ESTRUTURAÇÃO AUTOMÁTICA DE CONSULTAS BASEADAS EM PALAVRAS-CHAVE

ÉVERLIN FIGHERA COSTA MARQUES

ESTRUTURAÇÃO AUTOMÁTICA DE CONSULTAS BASEADAS EM PALAVRAS-CHAVE

Proposta de tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Orientador: Altigran Soares da Silva

Manaus

Novembro de 2011

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A Web é um grande repositório de informação que apresenta crescimento acelerado e sobre o qual usuários especificam consultas buscando informações sobre pessoas, produtos e serviços. Em muitos casos, estas informações estão não somente em documentos textuais, como por exemplo, páginas HTML ou blogs, mas também em bancos de dados on-line. Se as consultas obtiverem boas respostas, compra-se algum produto ou acha-se uma informação em pouco tempo. Portanto, a habilidade de dar boas respostas às consultas dos usuários é uma vantagem estratégica para máquinas de busca, sites de compras, redes sociais, etc.

Toda essa informação disponível na Web é recuperada principalmente por consultas especificadas usando palavras-chave. Assim, a busca por palavras-chave é bastante difundida entre usuários, principalmente porque é o que as máquinas de busca oferecem. É também um recurso de interface simples e intuitivo para acessar diversos repositórios de informação na Web e não requer conhecimento específico por parte dos usuários. Entretanto, o volume de dados na Web e a heterogeneidade de suas fontes de dados dificultam a obtenção de respostas precisas para consultas especificadas usando um pequeno número de palavras-chave.

Nas máquinas de busca típicas da atualidade, as interfaces de busca baseiam-se no modelo *bag-of-words* onde se assume que consultas e documentos são compostos por palavras e estas são independentes entre si. No entanto, trabalhos na literatura recente tais como Sarkas et al. [2010]; Tan & Peng [2008] advogam que, quando se deseja uma busca mais precisa, é necessário entender a intenção do usuário além do modelo *bag-of-words*. Para Tan & Peng [2008], o usuário pensa em conceitos ao formular as consultas e quando ele serializa os conceitos escrevendo-os através de palavras-chave, a individualidade dos conceitos se perde. Então, pode-se aplicar um processo de estruturação de consulta para recuperar essa individualidade. O ideal seria recuperar o relacionamento

estrutural entre as palavras da consulta automaticamente, tentar identificar quais palavras capturam conceitos para confirmar proximidade semântica entre
palavras ou restrições de ordem entre elas, e melhorar a qualidade das respostas obtidas. Uma grande dificuldade em inferir semântica diretamente da consulta é que, em geral, usuários utilizam poucas palavras (2 a 3) para especificar suas consultas e estas não ocorrem em uma ordem específica.

O processo de estruturação da consulta do usuário consiste em dividir a consulta em partes chamadas *segmentos* tal que cada segmento possa mapear um conceito semântico. Por fim, cada segmento recebe um rótulo associado ao conceito. Esse método também chama-se segmentação de consulta e é descrito por Tan & Peng [2008]. Estruturar consultas pode ajudar a melhorar as respostas obtidas pois, segundo Guo et al. [2009], a consulta dá boas pistas do conceito que o usuário está buscando.

Um cenário em que a estruturação de consultas é particularmente importante, é o das chamadas máquinas de busca verticalizadas, as quais, em geral, processam consultas sobre dados estruturados armazenados diretamente em bancos de dados. Em tal cenário, é útil estruturar a informação presente nas consultas e representá-la de acordo com um formato estruturado, como descrito em Li et al. [2009] e em de Sá Mesquita et al. [2007]. Um exemplo deste cenário são os sites de busca por produtos. Para lidar com esse contexto, Li et al. [2009] investigam a utilização de índices de bancos de dados. Por conter informação altamente estruturada e de natureza pouco ambígua para aplicá-la na estruturação das consultas, espera-se que os índices possam melhorar os resultados das buscas.

Métodos para estruturação automática de consultas escritas usando palavras-chave têm sido propostos em vários trabalhos na literatura recente. Porém, a semântica subjacente associada às consultas varia com o contexto. Bendersky et al. [2009] propõem um método para estruturar consultas com base na identificação de locuções substantivas (*noum phrases*); no trabalho de Yu & Shi [2009], a estruturação é feita com base em valores de atributos presentes em registros de um banco de dados; Guo et al. [2009] propõem a identificação e rotulação de entidades presentes na consulta; Li et al. [2009] definem um conjunto de rótulos fixos para anotar consultas no treino do método CRF (*Conditional Random Fields*); Tan & Peng [2008] assumem o conceito como unidade básica para estruturação e exploram o modelo de linguagem *unigram* usando o conceito como um *gram*.

O objetivo do trabalho aqui proposto é tratar de maneira uniforme o problema da estruturação de consulta de usuário em diferentes cenários de modo que os segmentos obtidos tenham boa qualidade semântica e representem precisamente a intenção do usuário. Nossos cenários de interesse são:

- máquinas de busca voltadas para consultas de produtos;
- sistemas de bancos de dados relacionais;
- bases de documentos estruturados e
- ambientes de busca facetada.

Então, precisamos desenvolver métodos ajustáveis à captura de evidências presentes em cada contexto e obter, por consequência, o melhor conjunto de rótulos. Julgamos que um conjunto de rótulos de qualidade é aquele que, entre todos os rótulos que possam ser associados a uma consulta, caracterize a totalidade dos termos significativos da consulta e que melhor reflita a intenção do usuário, o que é um dos desafios para a tarefa de estruturação de consultas.

Também é necessário que tais métodos sejam exequíveis diante do volume de informações a serem tratadas em cada contexto.

Ademais, é importante que os métodos mantenham-se escaláveis diante da variação do número de consultas de entrada.

Nossa proposta apresenta um método não-supervisionado para estruturar consultas de palavras-chave que usando informações estruturadas e de log para gerar um conjunto de rótulos de qualidade. Essas duas fases foram chamadas de rotulamento livre de ordem e rotulamento com ordem, respectivamente.

O método realiza o rotulamento em 3 passos e considera *features* de conteúdo e de estrutura para auxiliar a obtenção de rótulos de qualidade.

O primeiro cenário que exploramos é a estruturação de consultas de produtos formuladas por usuários em máquinas de busca. Obtivemos nesse cenário um ganho em relação ao *baseline* usando apenas informação do banco de dados para segmentar as consultas. Na segunda fase, usamos informações de log.

O restante desta proposta é dividido da seguinte forma: no Capítulo 2 descrevemos métodos de estruturação de consultas apresentados recentemente na literatura; o método de extração é apresentado no Capítulo 3; no Capítulo 4, descrevemos os experimentos realizados com as configurações no método. Por fim, apresentamos as conclusões e próximos trabalhos no Capítulo 5.

Capítulo 2

Revisão da Literatura

Podemos encontrar abordagens bastante distintas nos trabalhos relacionados sobre estruturação ou estruturação de consulta. As abordagens que empregam técnicas de aprendizado geralmente elegem um conceito que funciona como a unidade básica para aprender modelos a partir de treinamento derivado de logs de consultas de banco de dados. Já outros trabalhos exploram modelos tradicionais da recuperação de informação.

O processo de estruturação de consulta consiste em dividir a consulta em segmentos tal que cada segmento deve mapear um conceito semântico. Esse processo ajuda a melhorar as respostas obtidas uma vez que as consultas de usuários dão boas pistas do conceito semântico que o usuário está buscando. Entretanto, pode-se verificar que estruturar consultas não é uma tarefa trivial pois fazer a inferência de conceitos semânticos diretamente das palavras das consultas, que costumam ser consultas curtas, não fornece bons resultados. De fato, não é possível garantir alguma ordem ou encadeamento semântico entre as palavras das consultas formuladas pelos usuário. Por essas razões, a inferência de semântica e o tratamento de ambiguidade, que tem consequências negativas no processamento e na satisfação das necessidades de informação dos usuários, são os desafios da estruturação de consultas.

Entre os métodos estabelecidos para a estruturação de consultas, encontramos duas maneiras de estruturar, se olharmos as unidades produzidas pelo processo. Na primeira maneira, a consulta de palavras de usuário é segmentada e em seguida rótulos são associados aos segmentos gerados. Por essa razão, às vezes pode-se encontrar o termo rotulamento para descrever a estruturação. A outra maneira envolve gerar segmentos sem a obrigação de gerar rótulos. Ambas formas de estruturar consultas aparecem em diferentes contextos.

Em máquinas de busca voltadas para consultas de produtos, onde resolver am-

biguidade de uma consulta é estratégico, a estruturação pode ajudar a encontrar o produto mais próximo à consulta formulada pelo usuário. No contexto de sistemas de bancos de dados, ambas formas de estruturar têm sido usadas para auxiliar na redução do espaço de busca de respostas para as palavras da consulta. Nas buscas facetadas, a estruturação auxilia a extração de metadados da fonte de informação, como uma página Web, para mostrá-los como facetas na navegação do usuário.

Por fim, identificamos diferentes abordagens para a estruturação. Os trabalhos de Li et al. [2009] e Yu & Shi [2009] utilizaram métodos de aprendizado supervisionado com CRF e, embora consigam bons resultados de acurácia de sentença, normalmente empregam alguma estratégia adicional ao aprendizado para a correção de rótulos. Nos trabalhos de Guo et al. [2009] e de Li et al. [2009], focados em máquinas de busca, destacamos o uso de informações de logs de consultas, o que pode ser desafiante por questões de processamento e de acesso a essas informações. Tan & Peng [2008] usaram um método não-supervisionado para gerar segmentos sem rotular com dados de logs de consultas de máquinas de busca, mas precisaram usar informações da Wikipedia para refinar os conceitos obtidos. Cortez et al. [2010] descrevem um método não-supervisionado para extração de informação para entradas de *strings* de bases textuais para gerar rótulos em 3 fases.

Na próxima seção, apresentamos de forma detalhada os trabalhos citados acima.

2.1 Identificação de Entidades Nomeadas

O trabalho de Guo et al. [2009] tem como objetivo identificar nas consultas formuladas pelo usuário entidades nomeadas, procurando também determinar a classe semântica dessas entidades. Para isso, propõe um método chamado NERQ (*Named Entity Recognition in Query*).

Para o contexto de máquinas de busca comerciais na Web, os autores adaptaram a técnica de reconhecimento de entidades nomeadas (*NER*), da área de processamento de linguagem natural, para estruturar consultas. Uma das contribuições deste método é uma abordagem para quantificar a probabilidade de uma entidade nomeada pertencer a uma classe. Isso diferencia esse trabalho de outros anteriores cujas saídas eram apenas binárias, ou seja, que apenas determinavam se a entidade pertence ou não a uma classe. Outro diferencial do trabalho é a possibilidade de que a entidade nomeada pertença a mais de uma classe.

Para isso, uma consulta q é expressa por uma tripla (e, t, c), onde e é uma entidade nomeada (como um título de Livro, Música, Filme ou Jogo) , t é contexto de e e a

classe de e é denotada por c.

Após identificar na consulta as palavras que correspondem uma entidade nomeada, as demais palavras são consideradas como contexto. Para os autores, o contexto depende somente da classe, mas não da entidade nomeada. Assim, o contexto t é representado pela notação $\alpha\#\beta$, onde α e β denotam palavras à esquerda e à direita da entidade nomeada, que é substituída por #. Os autores também definem $G(q)$ como o conjunto de todas as triplas possíveis que representam uma consulta q.

Dada essa notação, citamos como exemplo a consulta *'harry potter walkthrough'*. Como *harry potter* é uma entidade nomeada, *walkthrough*, à direita da entidade nomeada, será o contexto. Logo essa consulta pode ser representada pela tripla ('harry potter', '# walkthrough',$*$), mas não por ('halo 3', '# walkthrough', $*$).

Dessa forma, após estabelecer a relação entre esses elementos, para uma consulta q, o método irá gerar todas possíveis triplas (e, t, c) e as que têm maior probabilidade entram no conjunto $G(q)$ para a consulta q. O resultado final do método surge após calcular todas as probabilidades $Pr(e, t, c)$ para todas triplas em $G(q)$.

O método NERQ consiste de 2 fases: *offline* e *online*. Na fase *offline*, algumas entidades nomeadas (como um título de Jogo, Música, Filme ou Livro) são selecionadas como sementes para serem rotuladas manualmente por humanos com as classes Jogo, Música, Filme ou Livro. Depois, o log de consultas é processado para achar sementes entre as consultas. Com esse conjunto de consultas, que contêm as entidades nomeadas sementes com seus contextos correspondentes, usa-se um método de aprendizado para descobrir as classes das entidades nomeadas. O próximo passo é estimar as probabilidades da classe dada a entidade nomeada $P(c \mid e)$ para cada entidade e a probabilidade do contexto dada a classe $P(t \mid c)$.

Após o aprendizado dos contextos das entidades, o log de consultas é processado novamente para coletar consultas que contêm esses contextos e o restante em cada consulta coletada é extraído como nova entidade nomeada. No fim da fase *offline*, espera-se todas probabilidades necessárias:

- a probabilidade da nova entidade nomeada, $Pr(e)$;
- da classe dada uma entidade nomeada, denotada por $Pr(c \mid e)$;
- a probabilidade de contexto para uma classe fixada $Pr(t \mid c)$

Na fase *online*, o método tenta achar todas as triplas (e, t, c) em $G(q)$ para uma consulta q. Então, constrói-se o conjunto $G(q)$ por: i) estruturação da consulta em entidades nomeadas e contextos com geração de todas possibilidades e ii) rotulamento das entidades nomeadas com todas as classes. Na sequência, para cada tripla (e, t, c)

em $G(q)$, calcula-se a probabilidade $Pr(e, t, c)$ e as triplas que tiverem as maiores probabilidades serão a resposta do processo. Portanto, espera-se, ao fim do método, a definição da classe semântica para a entidade nomeada.

Nos experimentos relatados no artigo, para as classes semânticas Livro, Música, Filme e Jogo, os autores selecionaram 180 entidades nomeadas dos sites Amazon, GameSpot e Lyrics como entidades nomeadas sementes. Depois, processaram as consultas de usuários existentes no log de consultas de uma máquina de busca Web comercial. Nesse log de consultas de usuários, havia 6 bilhões de consultas, onde 930 milhões consultas eram únicas. Com as entidades nomeadas sementes, extraíram 432.304 contextos e indexaram cerca de 1,5 milhão de entidades nomeadas. Para fins de avaliação, usou-se a métrica de precisão para os grupos de 25, 50, 100, 150 e 250 melhores resultados. Na média entre as classes, a precisão se manteve em 100% até os 50 melhores resultados. O menor valor de precisão, na média das classes, foi de cerca de 96% quando consideraram os 250 melhores.

Cabe destacar ainda o que os experimentos descritos nesse trabalho revelaram sobre as consultas de usuário:

- Mais de 70% delas possuíam entidades nomeadas;
- A maioria das consultas continham apenas uma entidade nomeada;
- Menos de 1% continham várias entidades nomeadas.

Esses dados enfatizam a importância de estruturar consultas de usuário para inferir semântica.

2.2 Rotulamento de Consultas usando modelos CRF

2.2.1 Derivação de Rótulos em Consultas sobre Produtos

Li et al. [2009] apresentam uma proposta para rotulamento de consultas de usuários no contexto de busca por produtos. Esse tipo de busca acontece, segundo os autores, em muitos domínios de dados estruturados e usualmente as informações que alimentam os sites vem de registros das tabelas de banco de dados. Essa característica pode favorecer a rotulamento de consulta através do uso dos metadados de banco de dados para resolver situações de ambiguidade. Por exemplo, uma consulta cujas palavras residem em um campo de uma tabela, o nome desse campo pode auxiliar no momento de rotular essa consulta.

Além disso, os autores acreditam que a ordem dos campos nas consultas de usuário revela padrões estatísticos importantes que auxiliam a rotulamento sequencial. Baseado nessa hipótese, escolheram o método CRF (*Conditional Random Fields*)Lafferty et al. [2001] para rotular as consultas por ser o estado-da-arte para tarefas de rotulamento de sequências. No caso deste trabalho, rótulos pré-definidos foram tipo, modelo, marca, atributo.

Porém, a aplicação do método CRF demanda a geração de instância de treino, ou seja, consultas rotuladas de alguma forma.

Como os autores explicam, se por um lado o processo de rotulamento manual das consultas de treino é caro e gerar um treino de tamanho adequado é inviável, por outro lado a obtenção não-supervisionada de rótulos, atinge taxas de erro inaceitáveis. Assim, eles utilizam uma estratégia conjugada com essas duas formas de treino para equilibrar as desvantagens citadas. Nessa estratégia, o treino do CRF foi dividido em dois grupos. O primeiro grupo foi extraído de log de consulta e rotulado manualmente. O segundo grupo envolveu uma maior quantidade de consultas que foram rotuladas automaticamente a partir de uma fonte externa. Esses rótulos do segundo grupo foram chamados de 'derivados' (*derived labels*).

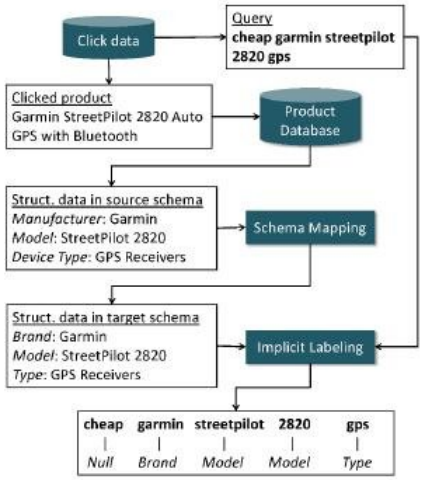


Figura 2.1. Arquitetura para obter rótulos derivados

A Figura 2.1 apresenta a arquitetura para geração dos rótulos derivados. Nesta figura, a consulta de usuário é a entrada. Assim, dada uma consulta como *'cheap garmin streetpilot 2820 gps'*, grava-se o evento de *click* na forma de um par (consulta,documento) no log de consulta. Interessam os *clicks* que levam a documentos que são páginas de listagem de produtos. Destas páginas se extraem os nomes dos produtos. Segundo os autores, isso é facilmente executado porque as páginas são bem

estruturadas.

Então, o evento de *click* pode ser representado na forma (consulta, título de produto). Nos casos em que este pares estiverem disponíveis em log de consulta, podem ser obtidos diretamente.

Na sequência, para os eventos de *click* (consulta, título de produto), procura-se no banco de dados por produtos com títulos similares, e são obtidos os metadados correspondentes, através de uma função de casamento aproximado. Depois, os metadados dos pares selecionados do banco de dados precisam ser mapeados para o esquema-alvo (marca, tipo, modelo, atributo). No fim da geração automática de rótulos, consideram-se os pares (consulta, metadados) e os metadados recém-mapeados para o esquema-alvo são 'corrigidos' de acordo com uma heurísticas sobre os tokens que ocorrem nos metadados:

- Se um token não aparece em nenhum campo ou se aparece em vários campos dos metadados, receberá rótulo *null*. Este é o caso de *'cheap'*, que não existe na base;
- Se um token aparece em exatamente um campo dos metadados, recebe o rótulo do nome do campo correspondente, por exemplo, *'streetpilot'* em Modelo.

Por fim, as consultas do log rotuladas manual e automaticamente são usadas para treinar o CRF na geração dos segmentos rotulados para as consultas do usuário.

Nos experimentos, os autores executaram o método CRF com os rótulos derivados em 2 configurações. Na primeira configuração, os rótulos manuais e derivados tiveram o mesmo peso no treino. Na segunda configuração, usaram os rótulos derivados como evidências adicionais combinando-os com *features* de *ngram*, de expressões regulares alfanuméricas e de léxicos com valores extraídos da base de produtos para Marca, Modelo, Tipo e Atributo. A última configuração obteve melhores resultados.

Li et. al. realizaram experimentos em duas categorias de produtos, roupas+sapatos e computadores+eletrônicos, e avaliaram os resultados medindo precisão na sentença, ou seja, o percentual de consultas corretamente rotuladas, e a precisão nos termos, ou seja, percentual de palavras com rótulos corretos.

2.2.2 Incorporando Preferências de Usuário

Yu & Shi [2009] tem como objetivos agrupar palavras vizinhas da consulta em segmentos e aprender o modelo estatístico das informações de logs de consultas.

A escolha de usar informações de logs deve-se à necessidade de incorporar preferências de usuário ao modelo. Além disso, deseja-se tratar a incorporação de de padrões (*patterns*) para incrementar a qualidade dos segmentos obtidos.

Por essas razões, os autores empregam o método CRF (Lafferty et al. [2001]) que é o estado da arte em tarefas de rotular sequências e permite

capturar uma grande variedades de *features* no modelo. Para aumentar as taxas de acertos nos resultados do método CRF, adotaram 2 modelos de CRF para a tarefa de estruturação. Em ambos modelos, o relacionamento entre rótulos e palavras da consulta é capturado por um conjunto de *features*. Além disso, adotaram a noção de segmento como unidade básica.

Dessa forma, modela-se y_{i-1} como rótulo da palavra anterior, y_i é o rótulo da palavra atual, x_{i-1} é a palavra anterior e x_i, a palavra atual. Essas variáveis vão gerar o conjunto $\mathbb{F} = \{f_1, f_2, \cdots, f_m\}$. As funções de *features* são expressas por $f_j(y_{y-1}, y_i, x_{i-1}, x_i, i) = \delta(y_{i-1} = L_1)\delta(y_i = L_2)\delta(x_{i-1} = w_1)\delta(x_i = w_2)$ onde L_i pertencem ao conjunto de rótulos , w_i são as palavras da consulta e $\delta(c)$ denota a função binária (1 se c é verdadeiro, senão 0). Na consulta 'Green Mile Tom Hanks', os autores exemplificam com a função de *feature* $f_7(y_{y-1}, y_i, x_{i-1}, x_i, i) = \delta(y_{i-1} = Actor)\delta(y_i = Actor)\delta(x_{i-1} =$'Tom'$)\delta(x_i =$'Hanks'$)$

Numa primeira fase, o conjunto de rótulos usados no processo de estruturação é retirado dos nomes das colunas do banco de dados. Na sequência, o método CRF computa os rótulos do conjunto para cada palavra numa dada consulta, baseado numa estruturação prelimiar que agrupou palavras vizinhas com mesmo rótulo em um mesmo segmento.

Seja uma consulta 'Green Mile Tom Hanks', os rótulos vêm do conjunto de colunas do banco de dados IMBD = $\{Filme, Ator, Companhia, ...\}$. Assim, a melhor sequência será
$\langle Filme, Filme, Ator, Ator\rangle$ e, então, o corte é feito $\langle\langle GreenMile\rangle\langle TomHanks\rangle\rangle$. Porém esse método gera um erro se a consulta for 'Johnny Deep Orlando Bloom'. Mesmo que o método gere corretamente os rótulos $\langle Ator, Ator, Ator, Ator\rangle$, estruturar $\langle\langle JohnnyDeepOrlandoBloom\rangle\rangle$ é um erro nesse caso porque esse termo de 4 palavras não existe na base.

A próxima etapa do primeiro estágio de CRF é particionar segmentos inválidos em válidos e identificar o segmento ótimo entre vários.

$$score((S)) = \sum_{S \in (S)} score(S) \tag{2.1}$$

Essa correção com a seleção do segmento ótimo máximo é feita por uma função de $score(S)$ que é definida para cada segmento S como mostra a Equação 2.1, onde $\mathbb{S}$ é o conjunto de todos segmentos ótimos.

A estruturação ótima que deseja-se encontrar é aquela que maximiza o score total

para todos segmentos, ou seja, $(S^*) = argmax\{score((S)\}$

Além de achar segmentos válidos e longos, é necessário minimizar os acessos ao banco de dados. Para tal fim, reduzindo custo de $(O^2(n))$ para $(O(n))$ acessos dada uma consulta de tamanho n, os autores propõe uma árvore para computar os segmentos ótimos para a fase de CRF 'melhorado'.

A segunda fase, chamada de CRF melhorado, foi adicionada à proposta pois nem sempre os resultados do CRF para tarefas de rotulamento de sequências são satisfatórios. Porém há a exigência de fazer a rotulamento e a estruturação em um passo único para melhorar a acurária dos resultados. Por essa razão, os autores melhoraram os resultados do CRF pela troca do conjunto de rótulos usados no treino e na inferência. Essa troca consistiu em adicionar a posição da palavras no segmento junto ao nome de coluna e ao rótulo no novo conjunto e logo, cada rótulo é um par posição-coluna. Entrentanto, para evitar uma explosão no tamanho do conjunto de rótulos e no número de função de *features*, apenas 'S' e 'N' apontam os rótulos de posição, onde 'S' indica o início do segmento. Também nessa fase, segmentos incorretos são gerados mas são corrigidos pela função de maximização de score.

Nos experimentos desenvolvidos, os autores mostram que esse modelo com 2 fases de CRF é superior aos métodos de limpeza de consulta e de busca gulosa ingênua. Também resolve com eficiência casos de consultas com muita ambiguidade. As métricas de avaliação foram acurácia de segmentos com inclusão de *features* de usuário.

2.3 Estruturação de Consultas usando Modelos de Linguagem Generativos

Para Tan & Peng [2008], quando o usuário formula uma consulta, ele pensa em conceitos. Porém, para especificar essa consulta usando uma interface típica de máquinas de busca na Web, é necessário serializar a consulta usando palavras e, por isso, a individualidade dos conceitos se perde.

Assim, os objetivos deste trabalho são recuperar as individualidades dos conceitos expressos pelas palavras da consulta e estabelecer relacionamentos estruturais entre estes conceitos de forma automática.

Este trabalho propõe uma abordagem estatística baseada em um modelo de linguagem, não-supervisionada, para capturar o processo de geração de boas sequências de segmentos que mapeiem conceitos completos e com as maiores probabilidades de casar com a consulta de usuário. Para isso, é utilizado um método de aprendizagem de máquina que busca evitar as desvantagens tanto dos métodos não-supervisionados,

que confiam demais nas estatísticas da coleção, quanto dos métodos supervisionados, cuja construção da coleção de treino é sempre muito trabalhosa e cara. Os autores também constataram que algumas abordagens anteriores focavam apenas nas dependências entre palavras e que não conseguiam capturar correlações entre mais de duas palavras ou tomavam apenas decisões locais para as segmentações.

A abordagem proposta neste trabalho usa a noção de *conceito* como unidade básica para estruturar a consulta. Sobre esses conceitos, assume-se a hipótese de que são independentes e identicamente distribuídos entre si, o que é chamado de *hipótese de IID*. Mesmo que isso não seja inteiramente verdade, essa hipótese permite que haja uma distribuição de probabilidade P_c de conceitos cujas amostras se repetem, para produzir conceitos mutuamente independentes que compõem uma consulta. Essa solução é, de fato, o modelo de linguagem *unigram*, usando o conceito como *gram*. De acordo com os autores, esse modelo é uma solução simples e também funciona razoavelmente bem para a tarefa de estruturar consultas.

Outra conveniência na hipótese de IID é que, se nos restringirmos somente às consultas curtas, a enumeração de segmentos através do modelo de *unigram* permite computar, por programação dinâmica, os top k melhores segmentos com complexidade logarítmica. Com essa forma de computar os segmentos adjacentes garante-se que a decisão da estruturação seja global.

Na sequência, o algoritmo de Máxima Expectância (EM) é executado *on the fly* numa parcela de coleção que é específica à consulta do usuário. Assim, o método resgata todas as partes da coleção que se sobrepõem à consulta de usuário para uma nova coleção e o algoritmo de EM segmenta os conceitos, em um primeiro passo. Depois, as probabilidades dos *n-grams* da consulta são computadas e servem como informação de estruturação para o segundo passo do algoritmo de EM. Os *n-grams* usados são as palavras adjacentes na nova coleção que foram concatenadas para formar *n-grams* mais longos com o intuito de resolver conceitos incompletos frequentes em coleções. A probabilidade do *n-gram*, na distribuição de conceitos, descreve qual a sua chance de aparecer num trecho de texto como conceito independente.

Embora essa forma de gerar segmentos otimize aspectos estatísticos dos conceitos, nenhuma consideração linguística é garantida ao obter conceitos na saída, isto é, os conceitos podem não ser bem formados.

Para corrigir esse problema, os autores usam a Wikipedia para guiar o processo de estruturação de consulta. Devido ao estilo de edição de conteúdo da Wikipedia, seus artigos têm uma alta cobertura de tópicos e esses tópicos são bem estabelecidos. Além disso, como os artigos são atualizados muito frequentemente, novos conceitos são introduzidos na terminologia da Wikipedia e a mantém dentro das tendências de

assuntos.

Essas características favorecem o uso dos atributos da Wikipedia como evidências adicionais para efetuar a correção dos segmentos. Depois de análise de atributos, os autores escolheram usar título do artigo, texto âncora e pontos de atalhos de alta frequência.

Nos experimentos realizados, usaram logs de busca da AOL usando como palavras de consultas apenas substantivos, adjetivos e artigos que tivessem algum resultado de *click*. Capturaram 464 milhões de *n-grams* de tamanho mínimo 5 e suas frequências. Para a avaliação, usaram as métricas de acurácia de classificação, de segmento e de consulta.

2.4 Extraindo Informação Sob Demanda

Cortez et al. [2010] desenvolveram um método não-supervisionado chamado Ondux para problemas de extração de informação. O método extrai estruturas implícitas em contextos onde as entradas de textos são livres e extensas, como em bases de endereços e referências bibliográficas.

O Ondux é um método de extração por estruturação de texto (*Information Extraction by Text Segmentation (IETS)*) aplicável a casos em que valores de dados de interesse são organizados em registros semi-estruturados implícitos em fontes de texto.

Assim como os métodos não-supervisionados de IETS, o Ondux se baseia em informação pré-existente, como valores de atributos que vem de fontes de dados, para associar segmentos nas entradas de *string* com atributo de um dado domínio. Esses valores de atributos são usados para formar as bases de conhecimento específicas de domínio (chamadas de *Knowlegde bases*).

Uma *Knowlegde Base*(KB) é um conjunto de pares $k = \{< a_1, O_1 >, \ldots, < a_n, O_n >\}$ onde cada a_i é um atributo distinto e O_i é um conjunto de *strings* $\{O_{i,1}, \ldots, O_{in_i}\}$ chamados ocorrências. Intuitivamente O_i representa o conjunto de strings que são valores típicos ou plausíveis para o atributo a_i.

A técnica distingue-se por efetuar a tarefa de extração em 3 fases: blocagem, *matching* e reforço.

Na fase de blocagem, as entradas de *strings* são divididas em *substrings* chamados blocos. O bloco é uma sequência de termos simples que formam um valor plausível para um dado atributo. Como em *'harry potter ps3'*, onde o conjunto de termos{ *'harry potter'*} deve pertencer a um bloco 'jogo'e { *'ps3'*}, a um bloco 'console'. Se um termo t_{j-1} e o próximo termo t_j coocorrem alguma vez na KB, t_{j-1} e t_j formarão o bloco,

senão um novo bloco recebe t_j.

No passo de *matching*, cada bloco gerado pela blocagem é associado a um atributo representado na KB. Para essa tarefa, o método usa um conjunto de funções de similaridade para efetuar o *matching* entre cada bloco contra as ocorrências que compõem a KB e assim determinar o atributo com maior probabilidade de representar um dado bloco.

A escolha da função específica para o *matching* do bloco ocorre após um teste simples que define o tipo de dados (como url, texto, numérico, email) dos termos que compõem aquele bloco.

Antes da fase de reforço, as entradas já estão rotuladas. Porém, a etapa de reforço ocorre justamente para revisar e corrigir quaisquer erros de rotulamento e quaisquer blocos que ficaram sem rótulos. No passo de reforço, constrói-se um Modelo de Sequenciamento e Posicionamento (PSM) com informação aprendida sobre os dados de teste sob-demanda(*on-demand*).

O PSM é um modelo probabilístico baseado em grafo e consiste de um conjunto de estados L, de uma matriz com probabilidades de transições T e de uma matriz com probabilidades de posições P.

O conjunto de estados do PSM é representado por $L = \{início, l_1, l_2, \ldots, l_n, fim\}$ em que cada estado l_i corresponde ao rótulo associado ao bloco obtido do *matching*, exceto para os estados *início* e *fim*;

A matriz T, do PSM, armazena a probabilidade de observar uma transição do estado l_i para o estado l_j na saída do passo de *matching* para o total de transições realizadas a partir do estado l_i.

A matriz P armazena a probabilidade de observar o rótulo l_i num bloco na posição k na saída do passo de *matching* para o número total de rótulos observados nos blocos que ocupam a posição k.

Na prática, as matrizes P e T são construídas em um passo único para maximizar as probabilidades de sequenciamento e posicionamento observadas para os valores dos atributos conforme os blocos rotulados na saída da fase de *matching*. Nesse ponto, blocos sem *matching* foram descartados ao construir essas matrizes. Ademais,Cortez et al. [2010] demostraram em experimentos que mesmo se blocos de *matching* ruim participaram da construção do PSM, as transições espúrias por eles geradas eram numericamente pequenas e não comprometeram a correção do modelo.

Além desse método possuir resultados comparáveis ao método CRF(Lafferty et al. [2001]), que é o estado da arte para tarefas de estruturação de consulta, a estruturação é feita em modo não-supervisionado e agrega, com o passo de reforço, refinamento dos rótulos resultantes.

Enquanto no passo de *matching* o bloco é formado de acordo com as funções de *matching*, apresentadas na seção 3.1, na fase de reforço, o PSM construído fornece as probabilidades de posicionamento e sequenciamento para as entradas de texto que vão participar na decisão de qual rótulo associar ao bloco. Para combinar as saídas de *matching* e com os dois tipos de probabilidades do PSM, os autores usaram um operador disjuntivo Bayesiano chamado *Noisy-OR-Gate*, tal como no trabalho de de Sá Mesquita et al. [2007], se estabelece a igualdade entre os fatores sem discriminar pesos.

Esse método obteve resultados equivalentes ao método do estado-da-arte dos algoritmos de extração porém, sem treino rotulado manualmente.

No Capítulo 3 descrevemos o método de extração empregado no trabalho.

Capítulo 3

Método de extração

Métodos não-supervisionados são mais apropriados em cenários onde há grande quantidade de dados disponíveis, como nas máquinas de busca, segundo Pound et al. [2011]. Esses métodos também foram explorados por Tan & Peng [2008] e Zhang et al. [2009] com uso de grande escala de informações de log para estruturar consultas. Embora esses autores tenham obtido resultados interessantes, não há geração de rótulos explícitos para as consultas.

Pelas razões comentadas, utilizamos aprendizagem não-supervisionada para extração de informação em *strings*. Adotamos os conceitos de blocagem, de *matching* e de reforço do método de Cortez et al. [2010] para nosso problema de estruturação de consultas de palavras-chaves.

Assim, para uma consulta de usuário, geramos blocos usando informações da base de dados para agrupar valores plausíveis da consulta para um dado atributo.

Na fase de *matching*, os blocos gerados recebem o rótulo do atributo da base que tem maior probabilidade de conter os valores do bloco. Essa atribuição de rótulos é efetuada por um conjunto de funções de similaridade que avaliam os blocos contra as ocorrências no banco de dados.

Por fim, na fase de reforço, o refinamento e a correção dos rótulos é feita com o aprendizado da ordem das informações de log de consultas. De fato, o log de consultas é a fonte de informação da qual serão aprendidas as probabilidades de posicionamento e sequenciamento do modelo PSM. Assim, o PSM é um conjunto de estados L, com as matrizes T de probabilidades de transições e P de probabilidades de posições.

O conjunto de estados do PSM é representado por $L = \{início, l_1, l_2, \ldots, l_n, fim\}$ onde cada estado l_i representa o rótulo associado ao bloco obtido do *matching*, exceto para os estados *início* e *fim*.

A matriz T contém as probabilidades de observar uma transição do estado l_i para

o estado l_j na saída do passo de *matching* para o total de transições realizadas a partir do estado l_i. Cada $t_{i,j}$ é dado pela Equação 3.1.

$$t(i,j) = \frac{\#\, de\, transicões\, de\, l_i\, para\, l_j}{Total\#\, de\, transicões\, que\, saem\, de\, l_i} \tag{3.1}$$

A matriz P contém as probabilidades de observar o rótulo l_i num bloco na posição k na saída do passo de *matching* para o número total de rótulos observados nos blocos que ocupam a posição k. Cada $p_{i,k}$ é dado pela Equação 3.2.

$$p(i,k) = \frac{\#\, de\, observacões\, de\, l_i\, em\, k}{Total\#\, de\, blocos\, em\, k} \tag{3.2}$$

Assim, para computar a função de atribuição do rótulo do bloco, também se assume:

- As probabilidades aprendidas pelo PSM são proveniente do log e são mutuamente indepententes.
- As funções de *matching* que representam a probabilidade de um bloco B ocorrer no valor do domínio do atributo a_i, de acordo com a base de dados e independem da fontes de dados.

Esses 2 fatores são independentes e podem ser combinados no operador *Noisy-OR-Gate* como
$or(x_1, x_2, \ldots, x_n) = 1 - ((1 - x_1) \times \ldots \times (1 - x_n))$,
sendo x_i uma probabilidade. Logo, escrevemos a função FS é escrita como Cortez et al. [2010].

$$FS(B, a_i) = 1 = ((1 - M(B, a_i)) \times (1 - t_{i,j}) \times (1 - p_{i,k})) \tag{3.3}$$

Na Equação 3.3, B representa o bloco encontrado na posição k em uma dada entrada de *string*, precedida por outro bloco conhecido associado ao atributo a_j. Os fatores ti, j e pi, k são as probabilidades armazenadas nas matrizes T e P, respectivamente. Por fim, a função FS é computada para cada bloco B na entrada de texto de todos atributos a_i com o mesmo tipo de dado e B receberá o rótulo do atributo que obtiver o maior score da função FS.

3.1 Features do método

Diferentes fontes de evidências têm sido usadas pelas técnicas de extração de informação para capturar valores de registros de dados semi-estruturados na forma de texto

contínuo.

Adicionalmente, em contextos onde existe uma camada de bancos de dados, o esquema do banco de dados é uma evidência valiosa para estimar *matching* entre entradas e valores de campos e tabelas, como mostraram de Sá Mesquita et al. [2007] e Sarkas et al. [2010].

Logo, essas fontes estruturadas e semi-estruturadas podem fornecer evidências valiosas de conteúdo ou de estrutura para a tarefa de estruturação de consultas. Das entradas de dados fornecidas por usuários, pode-se aprender *features* de conteúdo, e das bases, *features* de estrutura.

A estrutura do método é descrita na Figura 3.1.

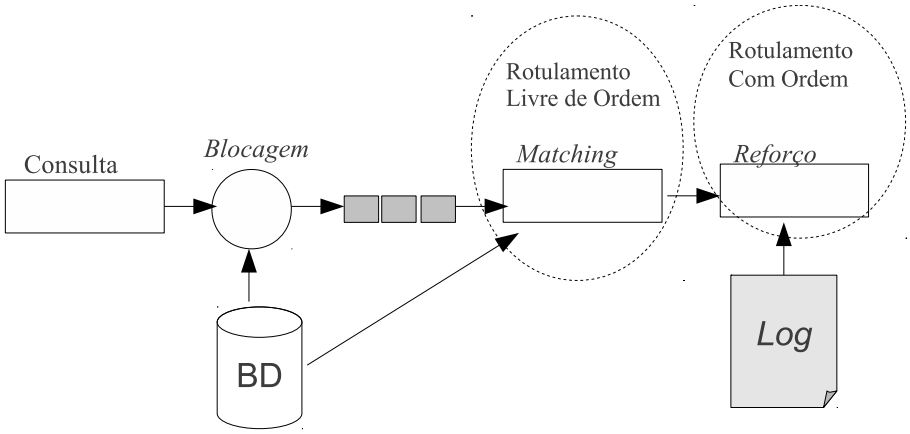


Figura 3.1. Método para estruturação de consultas

O aprendizado desses tipos diferentes de *features* deve auxiliar na descoberta de rótulos de qualidades.

3.1.1 Features Relacionadas ao Conteúdo

Essas *features* são dependentes de domínio e se pode obtê-las de fontes de dados pré-existentes como tabelas de relacionamento, bases de conhecimento, conforme comentaram Cortez et al. [2011]. *Features* de conteúdo podem ser capturadas do vocabulário do atributo, da escala de valores do atributo e do formato do valor do atributo.

- *Vocabulário do atributo* Essas *features* exploram o vocabulário comum compartilhado entre atributos do tipo texto. Para capturar esse propriedade, Cortez et al. [2011] adaptaram a função de atributo de frequência AF de de Sá Mesquita et al.

[2007]. A função AF é dada pela Equação 3.4.

$$AF(s, A) = \frac{\sum_{t \in T(A) \cap T(s)} fitness(t, A)}{|T(s)|} \tag{3.4}$$

A Equação 3.4 estima a similaridade entre o conteúdo de um valor candidato s e os valores de um atributo A representado na base de dados. Assim, $T(S)$ é o conjunto de todos termos encontrados nos valores do atributo A representado do banco de dados e $T(s)$ é o conjunto dos termos do valor candidato s. A função $fitness(t, A)$ avalia quão frequente é o termo t entre os valores do atributo A e é computado pela Equação 3.5.

$$fitness(t, A) = \frac{f(t, A)}{N(t)} \times \frac{f(t, A)}{f_{max}(A)} \tag{3.5}$$

Na Equação 3.5, $f(t, A)$ representa o número de valores distintos de A que contém o termo t, $f_{max}(A)$ é a frequência de qualquer termo entre os valores de A e $N(t)$ é o número total de ocorrência do termo t em todos atributos representados no banco de dados.

- *Escala de valores de atributo* No caso de valores candidatos numéricos (como preço, número de páginas, volume), funções de similaridade textual não funcionam adequadamente. Por tal motivo, para tratar valores candidatos dessa natureza, uma função adequada a essa *feature* é necessária. Como Agrawal et al. [2003] e Cortez et al. [2011], adota-se uma função que assume uma distribuição gaussiana para os valores numéricos dos atributos. Por assumir distribuição gaussiana para tais valores, a similaridade é estimada entre um valor numérico v_s presente no valor candidato s e no conjunto de valores v_A de um atributo A na banco de dados, avaliando quão próximo v_s está do valor médio de v_A de acordo com a função densidade de probabilidade. A função desse conceito é expressa na Equação3.6.

$$NM(s, A) = e^{-\frac{v_s - \mu}{2\sigma^2}} \tag{3.6}$$

onde σ e μ representam, respectivamente, o desvio padrão e a média de valores em v_A e v_s é o valor numérico que compõe s. Por consequência, a Equação 3.6 implica que a medida que v_s se aproxima da média dos valores em v_A, $NM(s, A)$ tende a 1. E quando v_s assume valores longe da média, a similaridade tende a zero.

No caso de valores numéricos em entradas de strings compostos também por caracteres especiais (como preços e números de telefone), é necessário remover tais caracteres e concatenar os números restantes. Esse processo chama-se normalização e deve acontecer antes da função NM ser aplicada.

- *Formato de valores de atributo* Outra propriedade a ser explorada é o formato comum frequentemente usado para representar valores de alguns atributos. Se v_A for o conjunto de valores disponíveis para um atributo A no banco de dados, automaticamente pode-se aprender a sequência do modelo de Markov m_A que captura o estilo de formato dos valores em v_A. Esse modelo pode capturar o formato dos valores como uma *feature* de estado. Para que isso aconteça, é necessário tokenizar cada valor de v_A em espaços brancos, aplicar a taxonomia proposta por Borkar et al. [2001] para codificar o valor como uma sequência de máscaras. As máscaras são identificadores da classe do caracter e podem ter quantificadores associados, como em 'Rivers inc.' que terá uma máscara '$[A-Z][a-z]+[a-z]+$ ' onde letras maiúsculas e minúsculas são representadas separadamente. Essa representação repete-se para todos valores conhecidos num dado atributo.

  Então, o modelo m_a é gerado usando essas codificações. No m_A, cada nó n representa uma máscara que ocorre nos valores de v_A. A aresta e entre os nós n_i e n_j existirá se n_i é seguida por n_j nas codificações. Por essa razão cada valor em v_A pode ser representado por um caminho em m_A. Para expressar a probabilidade de sequências de máscaras no modelo, definimos o peso da aresta $\langle n_x, n_y \rangle$ como $w(n_x, n_y) = \frac{\#de\ pares \langle n_x, n_y \rangle em\ m_A}{\#de\ pares \langle n_x, n_z \rangle, \forall n_z \in m_A}$

  Agora, seja s um valor candidato, codificamos s usando a taxonomia de símbolo como foi citado. Isso resulta em uma sequência de máscaras. Então, avaliamos quão similar um valor candidato s é dos valores em v_A em relação ao formato computando a Equação 3.7

$$format(s, A) = \frac{\sum_{\langle n_x, n_y \rangle \in path(s)} w(n_x, n_y)}{|path(s)|} \tag{3.7}$$

  Na Equação 3.7 $path(s)$ representa o caminho formado pela sequência de máscaras geradas por s em m_A. Note que se nenhum *matching* de caminho ocorrer para essa sequência em m_A, $format(s, A) = 0$. Intuitivamente, $format(s, A)$ informa quanto semelhantes são as sequências de símbolos formando um dado valor candidato s em relação às sequências de símbolos que tipicamente correm nos valores de um dado atributo A. Com essa *feature* é possível capturar propriedades específicas de formato de e-mails, URLs, por exemplo.

3.1.2 Features Relacionadas à Estrutura

Esses tipos de *features* são aprendidas de entradas fornecidas por usuários como arquivos de treino e são dependentes da fonte, como comentaram Cortez et al. [2011]. Os métodos do estado da arte em IETS geralmente usam dois tipos de *feature* de estrutura. Um primeiro tipo considera a posição absoluta do segmento de texto ou token que vai ser avaliado. O segundo tipo de *feature* considera sua posição relativa, por exemplo, sua ocorrência entre outros segmentos ou tokens no texto de entrada. Essas *features* são computadas usualmente por modelos de grafos para representar a probabilidade de transições das entradas de *strings*.

Enquanto nos métodos baseados em CRF se constrói o modelo com os dados de treino, que consiste de um subconjunto da entrada onde os registros foram rotulados manualmente, o modelo baseado em grafo é construído de forma não-supervisionada durante o próprio processo de extração.

Então, Cortez et al. [2010] propõem construir um modelo de grafo similar ao *Hidden Markov Model* chamado PSM (*Positioning and Sequencing Model*) que foi descrito na Seção **??**.

3.2 Usando log de consultas de usuário

Para capturar as *features* de estrutura, usamos log de consultas de usuário de uma máquina de busca por produtos. Com essa fonte de dados, pode-se aprender as probabilidades de posicionamento e sequenciamento dos blocos já existentes. Dessa forma, a construção o PSM vai prover evidências adicionais da ordem aprendida nas consultas postadas pelos usuários para correção dos rótulos obtidos após a fase de *matching*. Cabe ressaltar que no processo o log utilizado nunca é rotulado manualmente.

Outro objetivo da proposta é verificar se a seleção de entradas de log de consultas que gera o PSM pode melhor a qualidade dos conjuntos de rótulos obtidos para as consultas. Para verificar essa hipótese, para cada consulta de usuário presente no log calcula-se, após o passo de *matching*, um *score* de qualidade dos blocos do registro. Por fim, apenas as consultas com os maiores valores de *score* vão ser mantidas no log para gerar o PSM em uma nova execução do método.

No Capítulo 4 detalhamos os experimentos realizados para o cenário de consultas de produtos em máquinas de busca.

Capítulo 4

Experimentos

Como experimento inicial, avaliamos o método não-supervisionado citado no Capítulo 3 para o cenário de consulta de produtos em máquinas de busca.

Apresentamos os experimentos da base de produtos do site de busca por produtos em duas configurações: sem log e com log.

4.1 Configurações

Inicialmente comparamos o método de extração com CRF semi-supervisionado para o cenário de máquinas de busca voltadas para produtos. Nosso experimento usou uma base de dados com quase 7.000 itens da categoria computadores e eletrônicos[1]. O conjunto de teste foi composto de 200 consultas de usuário sem estrutura retiradas dos log de consulta de usuário.

Na avaliação dos resultados, comparamos os rótulos obtidos para o conjunto de teste com os rótulos do conjunto de referência e reportamos a taxa de acerto pelas métricas de Precisão, de *Recall* e de Medida-F.

4.2 Baseline

Utilizamos como *baseline* o método descrito em Li et al. [2009] para o problema da estruturação de consultas. Os autores empregam uma estratégia de treino onde consultas de usuário são rotuladas manualmente e a correção automática dos rótulos é feita com informação obtida de log de consultas. Entre as configurações de CRF utilizadas, a

[1]Formatada para o esquema (Tipo,Modelo,Marca,Atributo)

configuração de melhor desempenho foi o CRF semi-supervisionado com 3 conjuntos de *features*: de *ngram*, de expressões regulares extraídas dos atributos da base e de lexemas extraídos da base. Ademais, nessa configuração, a correção automática de rótulos é considerada como fonte de evidência adicional no processo de estruturação.

Os experimentos foram realizados com itens da categoria de computadores e eletrônicos. O treino foi feito usando 15 mil consultas manualmente rotuladas e 700 consultas para teste. Em Li et al. [2009], tanto o log quanto a base provêm da máquina de busca de produtos Bing. Para o processo de estruturação usaram 250 mil de pares (consulta, título de produto) que representam dados de *click* extraídos do log de consulta. Os resultados do *baseline* são reportados na Tabela 4.2 para um treino de 3 mil consultas rotuladas manualmente.

Tabela 4.1. Resultados de acurácia de consultas e de palavras do *baseline* com o treino de 3 mil consultas manualmente rotuladas.

# Consultas rotuladas automaticamente	ngram+regex+lex	
	Acurácia de Consulta	Acurácia de Palavra
0	68.31	84.10
20	68.01	84.87
200	69.96	85.22
2 mil	72.20	86.28
20 mil	72.05	86.00

4.3 Rotulamento livre de ordem

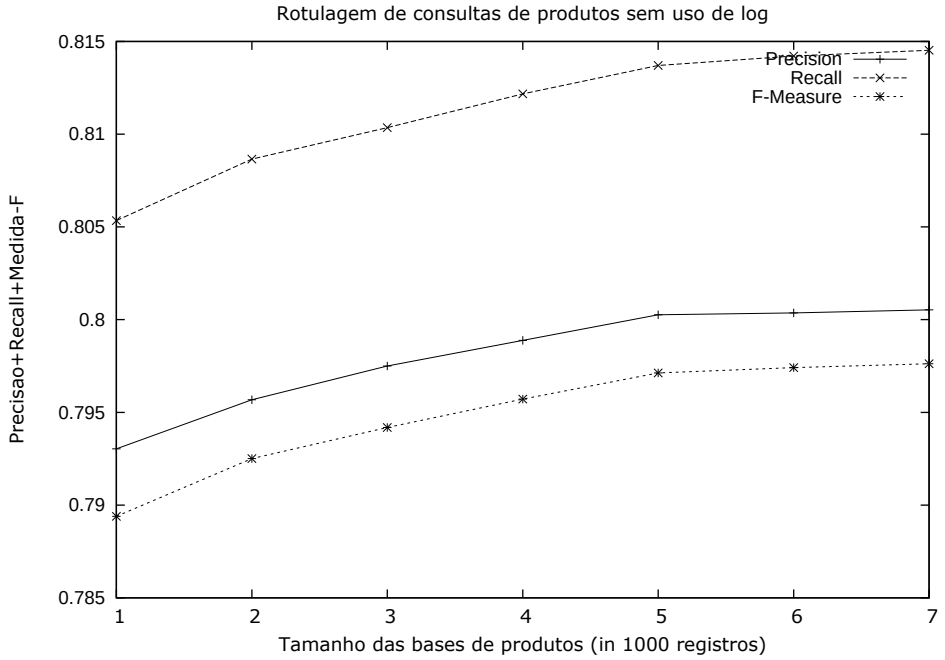
Executamos essa configuração e variamos o tamanho da base de registros para observar Precisão, *Recall* e Medida-F. Para tal, repetimos o experimento com o mesmo conjunto de teste a ser rotulado e variamos o tamanho da base sorteando os registros da base de produtos completa. Nessa etapa, não usamos quaisquer informações de log de consultas de usuário e obtivemos o gráfico da Figura 4.1.

A Figura 4.1 mostra que o método não-supervisionado utilizado com a estrutura da base e sem informação de log tem ganho crescente à medida que o tamanho da base aumenta. As amostras foram incrementados em 1.000 registros. A maior amostra contém 7.000 registros, onde cada registro representa um produto.

Comparamos os resultados dessa configuração aos resultados do *baseline* mostrado na primeira linha da Tabela 4.2, onde não há correção automática dos rótulos, mas somente o treino de 3 mil consultas de usuário rotuladas.

Enquanto no *baseline* reporta-se acurácia de cerca de 68% para as consultas corretamente rotuladas sem uso de consultas rotuladas automaticamente (da Tabela 4.2) e

Figura 4.1. Rotulamento de consultas de produtos por método não-supervisionado e sem uso de informação de log.



com a melhor combinação de *features*, obtivemos entre 80% e quase 82% de medida-F sem realização de treino.

O trabalho de Li et al. [2009] ainda reporta acurácia de 84% com 0 (zero) consultas rotuladas automaticamente, como mostra a Tabela 4.2. Então, o gráfico de acurácia de palavras do nosso método é mostrado na Figura 4.2 e reporta acurária superior a 86%.

4.4 Rotulamento com ordem

Na etapa em que executamos o método de extração com informação de log, usamos 6 mil consultas de usuário selecionadas com itens da categoria computadores e eletrônicos. Selecionamos apenas consultas com, no mínimo, 2 termos. Essas entradas serviram para aprendermos a ordem dos termos e gerar as probabilidades de sequenciamento e posicionamento do PSM na fase de reforço. Nessa fase de experimentos, o conjunto de produtos usado sempre foi de 7 mil produtos. Assim, com essa base, variamos o tamanho das consultas do log e reportamos a precisão, *recall* e medida-F.

Como o PSM construído na fase de reforço tem a finalidade de melhorar ou mesmo corrigir os rótulos obtidos da etapa de *matching*, era preciso garantir que o PSM fosse construído com as consultas mais representativas em relação à coocorrência dos termos.

Figura 4.2. Métrica de Acurácia de Palavra

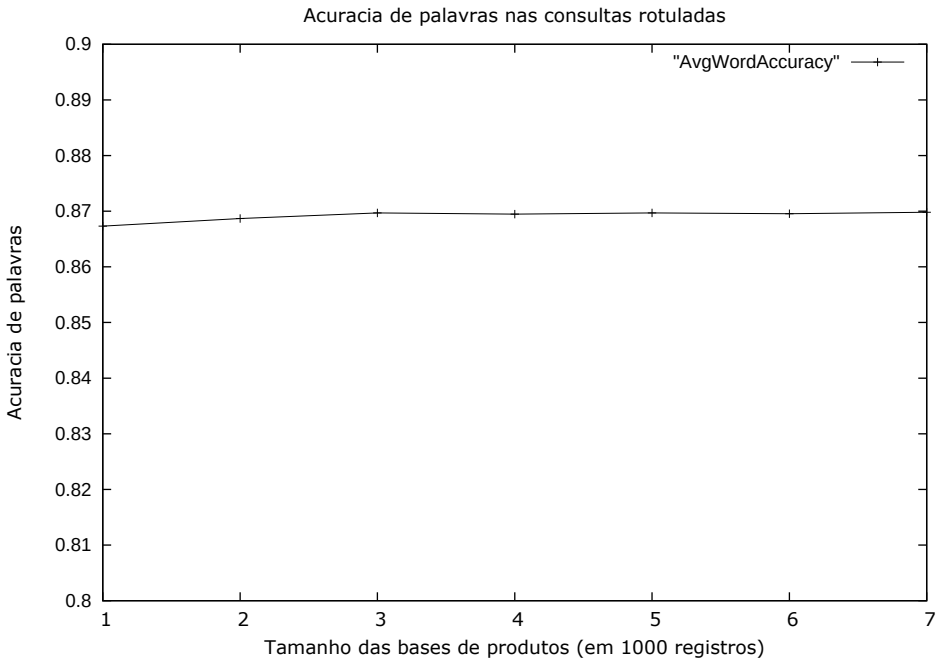
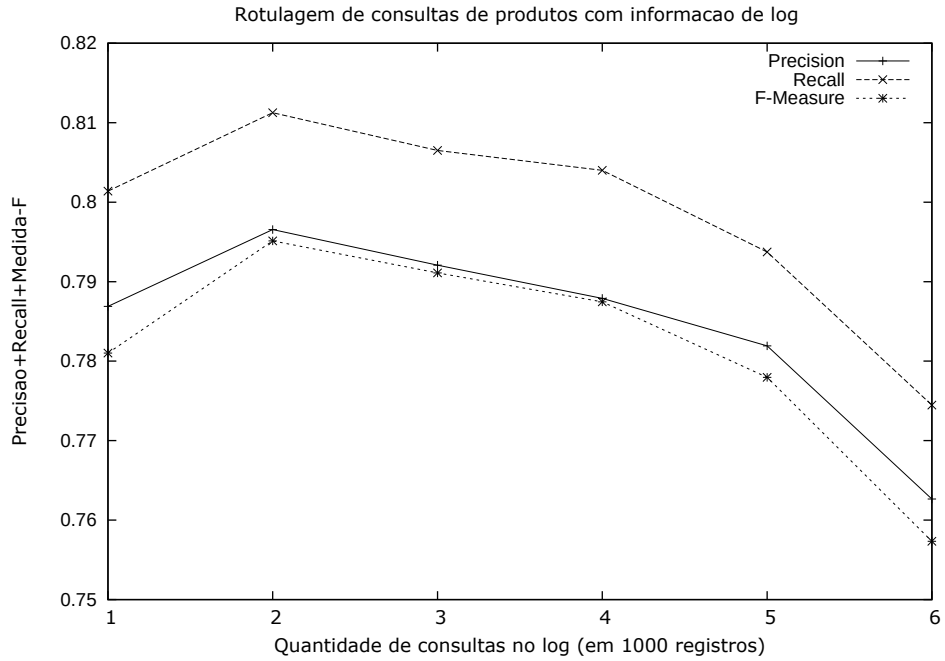


Figura 4.3. Rotulamento de consultas com o uso de informação de log (1000 a 6000)



Por exemplo, consultas com erros de digitação que recebessem um conjunto de rótulos menos frequentes, deveriam ser penalizadas antes de participar da construção do PSM.

De fato, esperamos conseguir melhoras nessa fase, com a seleção de registros do log para

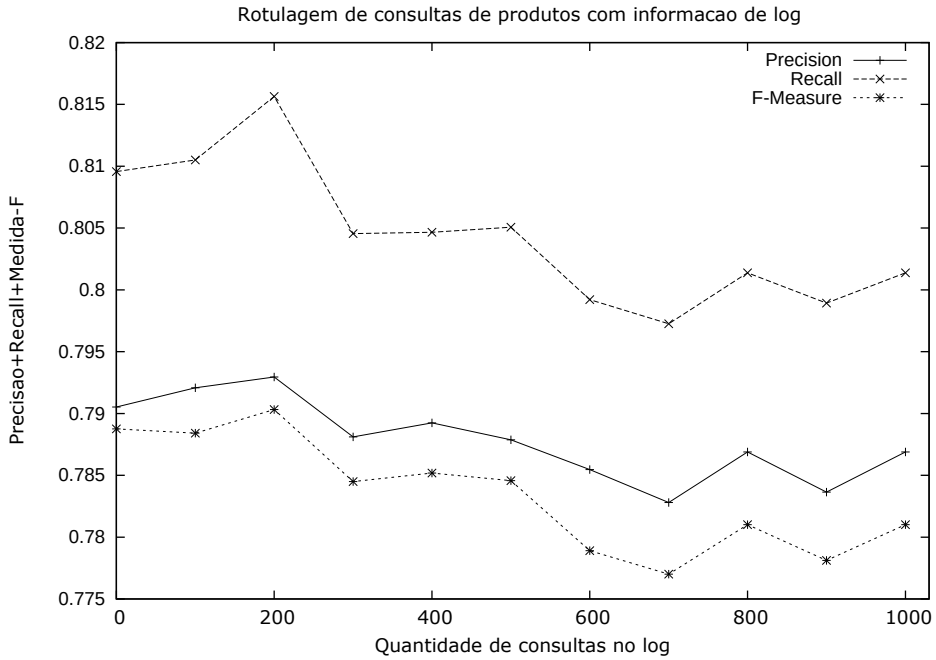
a construção do PSM. Para efetuar essa seleção, as consultas do log foram rotuladas como um conjunto de teste. Depois, essas consultas foram ordenadas por um *score* que media a qualidade do melhor conjunto de rótulos da consulta.

Esse *score* considera para o cálculo fatores como o melhor conjunto de rótulos, o número de termos por bloco, tamanho da consulta e o número de atributos do esquema (como Tipo, Marca, Modelo, Atributo).

Por essa razão, a curva decrescente da Figura 4.3 exprime que as menores amostras de log fornecem melhores evidências para a estruturação. À medida que mais registros do log ordenado são usados, a qualidade da estruturação cai, isso ocorre porque usar uma quantidade maior de registros do log ordenado traz consultas com conjuntos de rótulos pouco representativos sobre a ordem dos termos das consultas. Isso pode representar a inserção de ruídos no modelo do PSM.

Depois, expandimos o gráfico da Figura 4.3 no intervalo de 0 a 1000 registros e obtivemos a curva ilustrada na Figura 4.4.

Figura 4.4. Rotulamento de consultas com o uso de informação de log (0 a 1000 registros)



Embora nessa fase o comportamento de queda de precisão foi como se esperava, dado o aumento de consultas com pior posição na ordenação gerada pelo *score*, os resultados não apresentaram melhora significativa em relação aos resultados sem uso de log. Nos gráficos das Figuras 4.3 e 4.4, o melhor resultado é equivalente ao melhor resultado da fase de rotulamento sem ordem. Assim devemos reavaliar como organizar os registros do

log para obter uma melhora em relação à fase de rotulamento sem ordem. Inicialmente devemos reavaliar o ganho do *score* usado para ordenar o log em outras bases.

Na próxima seção, apresentamos as conclusões gerais da proposta.

Capítulo 5

Conclusões e Trabalhos Futuros

O problema da segmentação de consulta é explorado na literatura porque oferece vantagens estratégicas e pode melhorar os resultados de consultas nos bancos de dados e na Web.

Buscamos um método de rotulamento de consultas de palavras-chave que seja unificado em diferentes cenários. Nosso primeiro experimento foi realizado em consultas para máquinas de busca voltadas para produtos.

Nosso método obteve, na fase de rotulagem livre de contexto, 81% de medida-F sem esforço de treino e sem uso de informações de log.

Na fase de rotulagem com ordem, os resultados não foram muito melhores que na fase de rotulagem livre. Necessitamos, portanto, avaliar a qualidade do *score* em outras bases para assegurar que a seleção de informação de log realmente seja vantajosa para o rotulamento de consultas nesse contexto.

Após rever essa etapa referente à forma de seleção das consultas de log, avaliaremos o método nos outros cenários citados no texto. Já sabemos que, em cada cenário, precisaremos identificar o que pode ser usado como log de consultas. Também sabemos que a mudança de contexto requer identificar que unidades de dados podem fornecer os rótulos.

Referências Bibliográficas

Agrawal, S.; Chaudhuri, S.; Das, G. & Gionis, A. (2003). Automated ranking of database query results. In *CIDR*.

Bendersky, M.; Croft, W. B. & Smith, D. A. (2009). Two-stage query segmentation for information retrieval. In *SIGIR*, pp. 810--811.

Borkar, V. R.; Deshmukh, K. & Sarawagi, S. (2001). Automatic segmentation of text into structured records. In *SIGMOD Conference*, pp. 175–186.

Cortez, E.; da Silva, A. S.; Gonçalves, M. A. & de Moura, E. S. (2010). Ondux: on-demand unsupervised learning for information extraction. In *SIGMOD Conference*, pp. 807--818.

Cortez, E.; Oliveira, D.; da Silva, A. S.; de Moura, E. S. & Laender, A. H. F. (2011). Joint unsupervised structure discovery and information extraction. In *SIGMOD Conference*, pp. 541--552.

de Sá Mesquita, F.; da Silva, A. S.; de Moura, E. S.; Calado, P. & Laender, A. H. F. (2007). Labrador: Efficiently publishing relational databases on the web by using keyword-based query interfaces. *Inf. Process. Manage.*, 43(4):983--1004.

Guo, J.; Xu, G.; Cheng, X. & Li, H. (2009). Named entity recognition in query. In *SIGIR*, pp. 267--274.

Lafferty, J. D.; McCallum, A. & Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pp. 282--289.

Li, X.; Wang, Y.-Y. & Acero, A. (2009). Extracting structured information from user queries with semi-supervised conditional random fields. In *SIGIR*, pp. 572--579.

Pound, J.; Paparizos, S. & Tsaparas, P. (2011). Facet discovery for structured web search: a query-log mining approach. In *Proceedings of the 2011 international conference on Management of data*, SIGMOD '11, pp. 169--180.

Sarkas, N.; Paparizos, S. & Tsaparas, P. (2010). Structured annotations of web queries. In *SIGMOD Conference*, pp. 771--782.

Tan, B. & Peng, F. (2008). Unsupervised query segmentation using generative language models and wikipedia. In *WWW*, pp. 347--356.

Yu, X. & Shi, H. (2009). Query segmentation using conditional random fields. In *KEYS*, pp. 21--26.

Zhang, C.; Sun, N.; Hu, X.; Huang, T. & Chua, T.-S. (2009). Query segmentation based on eigenspace similarity. In *ACL/AFNLP (Short Papers)*, pp. 185–188.