RECODIFICAÇÃO DE ATRIBUTOS PARA LEARNING TO RANK USANDO AUTOENCODERS

ALBERTO DE SÁ CAVALCANTI DE ALBUQUERQUE

RECODIFICAÇÃO DE ATRIBUTOS PARA LEARNING TO RANK USANDO AUTOENCODERS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais - Departamento de Ciência da Computação. como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Renato Antônio Celso Ferreira
Coorientador: Adriano Alonso Veloso

Belo Horizonte

Janeiro de 2017

© 2017, Alberto de Sá Cavalcanti de Albuquerque
Todos os direitos reservados

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Albuquerque, Alberto de Sá Cavalcanti de

A345r Recodificação de atributos para learning to rank usando autoencoders / Alberto de Sá Cavalcanti de Albuquerque — Belo Horizonte, 2017.
xxviii, 67 f.: il.; 29 cm.

Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Renato Antônio Celso Ferreira.
Coorientador: Adriano Alonso Veloso.

1. Computação – Teses. 2. Recuperação da informação. 3. Aprendizado de máquina. 4. Aprendizado de ranqueamento. 5. Aprendizado de representações. I. Orientador. II. Coorientador. III. Título.

CDU 519.6*82(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Recodificação de atributos para learning to rank usando autoencoders

ALBERTO DE SÁ CAVALCANTI DE ALBUQUERQUE

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. RENATO ANTÔNIO CELSO FERREIRA - Orientador
Departamento de Ciência da Computação - UFMG

PROF. ADRIANO ALONSO VELOSO - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. EDLENO SILVA DE MOURA
Departamento de Ciência da Computação - UFAM

PROF. LEANDRO BALBY MARINHO
Departamento de Sistemas e Computação - UFCG

PROF. NIVIO ZIVIANI
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 23 de junho de 2017.

Aos meus pais, que sempre me incentivaram a olhar para o alto com os pés no chão.

Agradecimentos

Agradeço sobretudo aos meus pais, Alberto e Marília, pelo constante e perpétuo apoio a mim e às minhas ambições. Agradeço pelos questionamentos, puxões de orelha e broncas que, desde que me entendo por gente, me ajudam ser o que sou hoje. Meus agradecimentos se estendem a toda minha família e às pessoas especiais que tenho e tive privilégio de ter presentes em minha vida. Foram, e são, uma ilha de normalidade louca num mundo sempre inconstante.

Agradeço aos meus orientadores, Renato e Adriano, pelas instruções, acompanhamento, dicas e paciência que tiveram comigo ao longo de todo esse percurso. O Renato, em especial, já é meu mentor há sete anos, e certamente ajudou a moldar o profissional que hoje sou.

Agradeço aos inúmeros colegas de laboratório, sala e trabalho que estavam lá para responder às minhas perguntas, sempre presentes e insistentes. Muito do que sei se deve a eles, e muito do que sou se inspira neles. Agradeço também ao grande grupo de amigos que vem se formando desde minha época de colégio. São pessoas únicas, amizades ímpares, que sempre me ajudaram a tirar o vapor dos ouvidos nos momentos de maior estresse.

Agradeço, por fim, aos mentores e mentoras que há quase uma década vem me ensinando a caminhar nessa minha jornada do berço ao túmulo. O mundo é grande e intrincado, mas cabe a nós fazermos dele fácil ou difícil.

"I do not fear computers. I fear the lack of them."
(Isaac Asimov)

Resumo

Neste trabalho, nós avaliamos de forma abrangente o impacto e potêncial que o aprendizado profundo de representações alternativas tem enquanto aprimorador de resultados em tarefas de ordenação de documentos. Nós utilizamos autoencoders empilhados para criar um conjunto de centenas de representações alternativas de diversas bases de dados, e as avaliamos sob a ótica do desempenho de diversos algoritmos tradicionais de aprendizado de ordenação de documentos. Em outras palavras, procuramos saber o quão fácil ou dificíl é aprimorar a representação dos dados, usando autoencoders, que esses algoritmos utilizam em suas tarefas de aprendizado, e o quão melhores tais representações podem ser. Utilizamos o autoencoder para percorrer o domínio de representações possíveis de forma uniforme, de modo a, também, procurar entender o quão útil o autoencoder é para tal tarefa. Vemos em nossas análises que é possível, embora difícil, aprimorar a representação dos dados de forma relevante, obtendo resultados superiores ao estado da arte nas tarefas de ordenação de documentos. Vemos também que há conjuntos de hiperparâmetros do autoencoder que tendem a gerar resultados melhores.

Palavras-chave: Aprendizado de Máquina, Recuperação de Informação, Representações, Aprendizado de Ordenação de Documentos.

Abstract

In this dissertation, we evaluate thoroughly the impact and potential of deep learning alternative representations as a way to improve results in Learning to Rank tasks. We use stacked autoencoders to create hundreds of alternative representations of several databases, and evaluate those under the prism of their impact in the performance of several traditional Learning to Rank algorithms. In other words, we ask how easy or hard it is to improve the representation of Learning to Rank data using autoencoders, and how better can these representations be. We use the autoencoder to uniformily walk the domain of possible representations, so to know as well how useful can the autoencoder be for such task as well. We see in our analysis that it is possible, although difficult, to improve the databases representations in a relevant way, obtaining results that surpass state-of-the-art performances of the traditional ranking algorithms. We see, as well, that there are autoencoder hyperparameter sets that tend to generate better results.

Keywords: Machine Learning, Information Retrieval, Representations, Learning to Rank.

Resumo Estendido

Introdução

Nesta seção falamos acerca do problema que enfrentamos e de sua relevância no mundo de hoje. Falamos sobre o estado atual da indústria, sobre onde estamos e sobre quais contribuições nos propomos a fazer.

Revisão Bibliográfica e Conceitos

Aqui discutimos longamente acerca do que existe na literatura em seleção de *features*, problema associado ao que pretendemos fazer, e do que há em Ordenação de Documentos, área onde nos propomos a atuar. Falamos acerca do que é o autoencoder e como o utilizaremos para solucionar o problema observado.

Aprendizado de Representações

Discorre-se aqui acerca da forma com que utilizaremos o autoencoder para atuar sobre o problema proposto, e o que esperamos descobrir.

Experimentos

Nesta seção, delineamos o conjunto de hiperparâmetros que utilizamos em nossos experimentos, e enumeramos os algoritmos de aprendizado de ordenação que utilizamos como avaliação. Em seguida nós discorremos acerca dos resultados que tivemos, de forma geral e específica: base por base, algoritmo por algoritmo, métrica de qualidade por métrica de qualidade. Fazemos então um apanhado de todas as nossas conclusões e discutimos acerca de perguntas advindas destas.

Conclusões

Aqui há a conclusão desta dissertação, com um resumo de tudo que fizemos e ampla discussão acerca de trabalhos futuros em potencial.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Nós vivemos na era da informação. Com o advento das redes sociais, avanço da tecnologia computacional e uma quantidade cada vez maior de sensores espalhados pelo mundo, temos como maior desafio a extração de informações relevantes desse universo de dados. Existem dados em excesso: temos várias formas de quantificar e modelar cada problema, e capacidade de processar petabytes em relativamente pouco tempo. O problema surge, entretanto, da maneira como se faz a análise das descobertas: existem dados relevantes, dados redundantes e muito ruído, e em tamanha quantidade que saber a priori o que levar em consideração é uma tarefa em muitos casos inviável. Criar modelos eficientes que representam com precisão o que se quer é um desafio não trivial [2, 29], e muitas vezes tentativas do tipo caem na maldição da dimensionalidade: como são dados demais sendo levados em consideração, é difícil separar o bom do ruim e o resultado final acaba severamente comprometido.

Por outro lado, saber separar razoavelmente bem esse joio do trigo é o que viabilizou a existência de tantos negócios nas últimas décadas. É possível encontrar em contextos como vendas, publicidade, redes sociais, criação de conteúdo, e em inúmeros outros, problemas resolvidos e soluções aprimoradas pela análise inteligente de informações. Num contexto de busca na web, por exemplo, ordenar sites por relevância sempre foi um desafio crucial para tornar uma ferramenta de busca útil para o usuário. O algoritmo de pagerank [31], nesse contexto, ajudou a melhorar tal ordenação de respostas analisando a maneira como os sites se interligavam através de hiperlinks. Esse é um exemplo típico de extração de informação útil, o dito mapa de links, de um mar de dados, todo o conteúdo dos sites da web. Viu-se, aí, uma relação entre o número de "conexões" que um site tinha com os demais com o tanto que ele seria útil para o usuário. O resultado foi uma grande melhora na obtenção de informação relevante para o usuário. Ao longo dos últimos 50 anos os estudos em Recuperação de Informação

levaram à criação de diversas métricas que estimam, com maior ou menor precisão, a relevância de um documento para uma consulta utilizando um conjunto de palavras chave, sendo TF-IDF [38] e BM25 [37] as mais importantes dentre várias. O conjunto dessas métricas de relevância extraídas de um documento é denominado metadados do documento.

Dentre as inúmeras formas de se fazer análise de dados de todo tipo que surgiram nas últimas décadas, temos o Aprendizado de Máquina. Arthur Samuel definiu essa área, em 1959, como sendo capaz de "dar aos computadores a habilidade de aprenderem sem serem explicitamente programados" [42]. A idéia básica por trás disso é analisar uma grande massa de dados em busca de padrões extraíveis que contribuam para um determinado tipo de análise, seja esta uma classificação, um agrupamento de entidades semelhantes, ou alguma outra coisa.

Learning to Rank é a área do Aprendizado de Máquina que estuda o uso das tais técnicas para se classificar um conjunto de documentos por ordem de relevância. Dados e metadados de grandes conjuntos de documentos são submetidos a uma análise de padrões que busca encontrar informações que, de forma explícita ou implícita, levem à determinação do grau de relevância que um dado documento tem para uma determinada consulta do usuário. Muitos trabalhos em Learning to Rank são voltados para a criação ou o aprimoramento de tais algoritmos de ordenação de respostas [6, 8, 36, 43, 48, 50], tendo como foco a busca por maneiras melhores de se extrair informações dos metadados. Em outras palavras, esses algoritmos buscam encontrar uma relação preditiva entre os metadados e a relevância de um documento.

Tendo isso em mente, surge o questionamento acerca da maldição da dimensionalidade: será que, havendo uma quantidade tão abundante de metadados, não existe nas bases de dados muito ruído que, ao invés de ajudar os algoritmos de ordenação de respostas, os atrapalha? A pré-seleção de metadados de documentos para uso otimizado nos algoritmos de ordenação de respostas, ou *seleção de atributos*, vem sendo usada há vários anos com sucesso para se aprimorar os resultados apresentados pelos algoritmos de ordenação de respostas. Trabalhos anteriores comprovam [20] que atributos em excesso podem atrabalhar a geração de resultados em análises de dados, uma vez que elas podem ser redundantes ou acrescentar pouca informação, resultando numa análise de padrões mais limitada. Fazer a seleção de atributos, no entanto, é um processo custoso que demanda cuidadosa análise da natureza dos dados em questão. Por mais que exista muito estudo acerca das melhores formas de se analisar os metadados das bases de dados de Learning to Rank, há espaço para estudos acerca do aprimoramento dos atributos que tais algoritmos usam.

Neste trabalho, nós utilizamos técnicas de aprendizado de representações para

fazer uma seleção automática de atributos de Learning to Rank. Isto é: nós propomos a utilização de técnicas de aprendizado de máquina para, de forma automática, criarmos representações melhores dos metadados das bases de dados de Learning to Rank. Nosso intuito foi analisar o potencial dessas técnicas enquanto aprimoradoras de resultados de ordenação de respostas e, mais do que isso, mostrar de que forma os diferentes algoritmos de ordenação de respostas são afetados por tais mudanças.

1.1 Objetivos

Nós utilizamos diversas variações de um tipo de rede neural, denominado autoencoder, para gerar um grande número de representações de várias bases de dados de Learn to Rank e avaliamos o desempenho dos algoritmos de ordenação de respostas tradicionais da área quando executados sobre essas novas bases que geramos, e comparamos tal desempenho com o das bases originais. Nós mostramos que há um conjunto reestrito de configurações de autoencoder capaz de gerar representações boas dos metadados, e que este conjunto é comum a todas as bases de dados estudadas. Vemos também que cada algoritmo de ordenação de respostas reage às novas representações de formas diferentes: uns tem o resultado muito aprimorado para a maior parte das bases, e outros tem melhora menor, e em menos bases. Vemos, por fim, que a diferença de desempenho dos algoritmos de ordenação de respostas é minimizada na medida em que as representações das bases melhoram. Ou seja, nós mostramos que: (1) é difícil, porém possível, melhorar a representação das bases de dados de Learning to Rank utilizando autoencoders, (2) há uma faixa estreita de parâmetros da rede onde, para as diversas bases estudadas, essa melhora ocorre e, (3) se a representação da bases de dados for boa o bastante, os diferentes algoritmos de ordenação de respostas tendem a ter desempenho semelhante.

1.2 Organização

No capítulo 2 nós exploramos os estudos em Aprendizado de Representações e em Learning to Rank que evidenciam o potencial do caminho que optamos por seguir. No capítulo 3 nós detalhamos a nossa contribuição, com a descrição da abordagem que que tivemos para a análise do problema. No capítulo 4 nós analisamos os resultados dos nossos experimentos, e no capítulo 5 nós apresentamos nossas conclusões.

Capítulo 2

Revisão Bibliográfica e Conceitos

2.1 Revisão Bibliográfica

A seleção de atributos é usada há algum tempo como forma de buscar melhorias nos resultados dos algoritmos de ranking, com resultados que indicam seu grande potencial [19]. O excesso de atributos pode contribuir para a piora dos resultados dos algoritmos de ordenação de respostas, seja pelo acréscimo de ruído das mesmas, pela redundância, ou pelo fato delas ocasionarem em overfitting, fenomeno que ocorre quando, num algoritmo de aprendizado, uma rede se adequa bem demais a um conjunto de treino, perdendo generalidade. As técnicas de seleção de atributos geralmente estão englobadas em 3 grupos: *filter, wrapper* e *embedded*. O método de *filter* computa um escore para cada atributo separadamente, e utiliza as melhores. O *wrapper* usa um algoritmo de busca para explorar o espaço de possibilidades dos subconjuntos de atributos, para então estimar a qualidade de cada grupo usando apenas os atributos deste no processo de aprendizado. Já os métodos *embbeded* são técnicas integradas no processo de aprendizado [20].

O algoritmo GAS, proposto em 2007 [19], é um algoritmo guloso que seleciona os atributos baseado na maximização da relevância e minimização da similaridade com os atributos escolhidas anteriormente. Tal algoritmo foi melhorado, com obteção de resultados ainda melhores, usando seleção por pares ou grupos de atributos [20]. Já Hua *et. al.* [22] criaram uma técnica onde o k-means é usado para agregar atributos similares, de modo que depois o atributo mais relevante de cada conjunto é escolhido. Outro algoritmo [33] usa árvores de regressão num algorimo guloso. Algoritmos *embedded* como [25, 26] selecionam atributos e constroem o modelo de ranking no mesmo momento.

Indo além da mera seleção de atributos, há a possibilidade de transformação

das atributos. Ao contrário da seleção de atributos, onde o objetivo é identificar um subconjunto de atributos que tenha eficácia comparável ou superior ao uso de todos os atributos, a idéia central por trás da transformação de atributos é criar atributos melhores pra representar os dados de entrada, usando os atributos originais como base. Em [14, 15], os autores propuseram uma abordagem onde dados não-classificados são utilizados papara se derivar padrões relevantes que geram novos atributos [39]. De modo semelhante, em [34] os autores propuseram o uso de tal abordagem para se fazer o aprendizado de funções de ranking. A intuição que descreve tal idéia é a de que busca-se obter dados de preferência pareados usando-se regras associativas a partir de dados de teste não classificados.

Técnicas de aprendizado de máquina já foram usadas para a transformação de atributos, como em reconhecimento de imagens [9, 21], apesar de pouco em Seleção de Atributos para Learning to Rank [1]. Em [41], uma rede convolucional é usada para criar novas representações de textos entrados, em busca de uma ordenação de respostas mais eficiente em Learning to Rank. Ao contrário dos algoritmos de seleção de atributos, no entanto, ainda há pouco estudo na área de transformação de atributos, apesar de forte evidência que indica que a forma com que os dados são apresentados pode influenciar enormemente o sucesso de um algoritmo [17].

2.2 Conceitos

Em 1996, foi proposta [30] uma técnica de representação comprimida de imagens, feita através da minimização do erro de reconstrução da imagem com uso de uma rede neural de dimensionalidade reduzida. Ou seja: busca-se reconstruir uma imagem da melhor forma possível utilizando uma quantidade de dados reduzida, quando comparada com a imagem original. Tal representação é construída com um autoencoder, uma rede neural construída para reconstruir os dados entrados utilizando como base uma ou mais camadas intermediárias com dimensionalidade reduzida. Essa técnica vem evoluindo com o passar do tempo [9], e vem mostrando resultados muito promissores [1, 21, 24].

Um autoencoder, em resumo, é uma rede neural formada por dois componentes: um codificador (encoder) e um decodificador(decoder). O codificador recebe a entrada $x = \{x_1, x_2, \ldots, x_n\}$ e a mapeia para a representação oculta $y = \{y_1, y_2, \ldots, y_m\}$, que é dada por:

$$y = \sigma(Wx + b) \tag{2.1}$$

onde b é o valor do bias, W é a matriz de pesos e σ simplesmente representa uma sigmóide que opera sobre cada elemento.

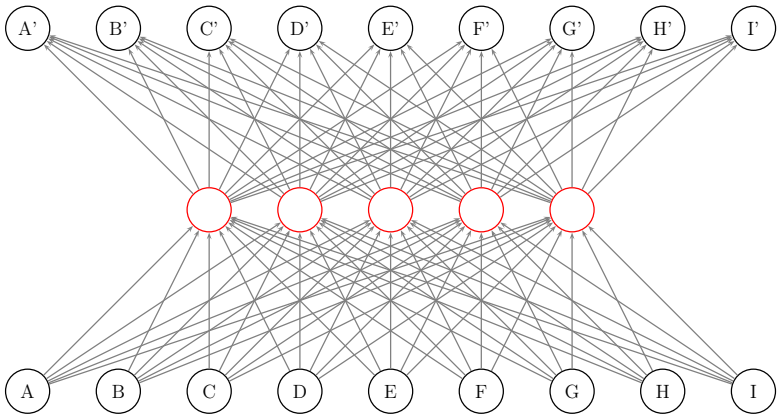

Figura 2.1. Um autoencoder.

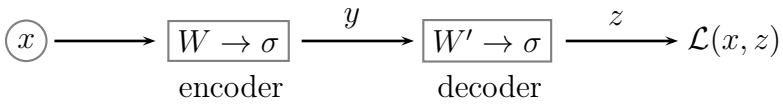

Figura 2.2. Aprendendo representações alternativas com um autoencoder

A representação oculta, ou latente, y é então mapeada para os dados de entrada originais, x, usando-se um autoencoder que é dado por:

$$z = \sigma(W'y + b') \tag{2.2}$$

onde b' são os valores e bias e W' é a matriz de pesos. No nosso contexto, W' não é necessáriamente a matriz transposta de W.

Autoencoders não supervisionados usam backpropagation para minimizar o erro de reconstrução, dado por:

$$\mathcal{L}(x, z) = ||x - z||^2 \tag{2.3}$$

onde as entradas com baixo erro de reconstrução são associadas com probabilidades maiores $P(x)$. Assim, os atributos de um autoencoder são produzidos num processo de maximização de $P(x)$.

A figura 2.2 mostra uma ilustração conceitual de um autoencoder. Muitos autoencoders podem ser empilhados e treinados em sequência − uma vez que um autoenco-

der inferior é treinado, o próximo autoencoder é treinado computando a representação oculta gerada pelo autoencoder inferior. A idéia é que cada autoencoder trabalha sobre a representação do autoencoder inferior, produzindo resultados de abstração cada vez maiores, por ser composto por mais operações. É possível também adicionar-se ruído aos dados de entrada do autoencoder [47]. Especificamente, a entrada x é corrompida com ruído, mas o autoencoder é treinado para recuperar a entrada original. Ao treinar para, na prática, desfazer o processo de corrompimento, a representação oculta y que o autoencoder gera se torna robusta a pequenas mudanças na entrada, que por menores que sejam poderiam impactar os resultados [13]. A junção desses três estilos de autoencoder resulta no chamado *stacked denoising autoencoder*.

A forma de treino do autoencoder pode ser não supervisionada, supervisionada ou semi-supervisionada. No treino não supervisionado, o autoencoder funciona na sua forma "padrão": ele, para cada dado que ele recebe de entrada, tenta aprender a reconstruir tais dados na camada de saída utilizando como base os dados da camada intermediária. Uma representação x, ou seja, é comparada com a representação x' gerada, tendo-se como base a camada intermediária, e busca-se minimizar esse erro de reconstrução. No treino supervisionado, por outro lado, é acrescentada sobre este treino uma camada extra de feedback: cada modelo, após ser recriado, deve ser associado pela rede a uma classe específica, previamente definida. Ao invés, portanto, de uma representação x ser associada a uma representação reconstruída x' somente, ela é associada a um conjunto de probabilidades $P(x)$, que representam a probabilidade da entrada pertencer a uma classe específica. A cada iteração deste treino busca-se minimizar o erro de classificação da rede, de forma que a etapa de minimização de erro de reconstrução é somente parte do processo de criação da rede. Autoencoders semi-supervisionados geralmente são construídos com uma mistura das duas técnicas anteriores.

A Recuperação de Informação é a área que estuda, entre outras coisas, técnicas de ranqueamento de documentos em grandes bases de dados e na web. Ao longo dos últimos 50 anos os estudos da área levaram à criação de diversas métricas que representam, com maior ou menor precisão, a relevância de um documento para uma consulta que utiliza um conjunto de palavras chave, sendo TF-IDF [38] e BM25 [37] as mais importantes dentre várias. No contexto de buscas na web, medidas como PageRank [31] aprimoram ainda mais os resultados. A web tem bilhões de páginas, e dada a quantidade de dados disponíveis e o acesso cada vez mais fácil a alto poder de processamento, há cerca de dez anos tecnologias de Aprendizado de Máquina vem sendo usadas em pesquisas voltadas para Recuperação de Informação, numa área denominada "Learning to Rank".

Uma grande quantidade de algoritmos de ranking automático vem sendo proposta, com divergências nas técnicas de ranking utilizadas e métricas de sucesso [28]. Tradicionalmente, em Recuperação de Informação, existem diversas formas de se determinar quais documentos tem maior relevância, e as métricas convêncionais se dividem em duas categorias:

As métricas denominadas *Query Dependent*, ou "dependentes da consulta", são aquelas cuja avaliação depende não só do documento em si, mas de sua relação com a consulta do usuário. Isso significa, naturalmente, que cada par consulta-documento produz um resultado diferente. TF-IDF [38] e BM25 [37] são as métricas mais tradicionais dessa área. No TF-IDF, a TF, a frequência do termo, é dada pelo número de vezes que o termo aparece no documento sendo avaliado, num valor normalizado. O IDF, a frequência invertida dos termos nos documentos, cresce em valor na medida em que o documento aparece em menos documentos. Isto é: quanto mais rara for a presença do termo da coleção de documentos, maior a relevância. O valor do TF-IDF é computado para cada um dos termos da consulta. O resultado final dessa análise é um número que representa a relevância do documento para aquela consulta, onde quanto maior, melhor.

O BM25, por sua vez, é toda uma família de métricas composta por variações de uma fórmula base comum. Ele é uma métrica probabilistica de relevância de um documento para uma determinada consulta, que usa os próprios TF e IDF em sua fórmula, assim como outros parâmetros estáticos obtidos empiricamente. Além destas e outras, temos métricas *Query Independent*, isto é, independentes de consulta, que buscam definir a relevância de um documento baseadas na sua importância própria. A mais famosa destas é o PageRank [31], que mede a probabilidade de um usuário, clicando em links aleatóriamente, acessar uma determinada página na web.

Em bases de dados de Learning to Rank, cada par consulta-documento está associada a uma métrica de relevância, que pode tanto ser binária (1 para relevante, 0 para irrelevante) quanto gradualizada (por exemplo, 2 é 'muito relevante', 1 é 'possívelmente relevante' e 0 é 'irrelevante'). Muitas vezes os documentos associados a uma mesma query são organizados em pares e enumera-se, para cada par, qual é o documento mais relevante. Diversas métricas foram propostas para se avaliar um algoritmo de ordenação de respostas de R.I., comparando seus resultados com uma benchmark classificada manualmente por seres humanos. Dentre elas temos *Mean Reciprocal Rank (MRR), Mean Average Precision (MAP), (Normalized) Discounted Cumulative Gain, Rank Correlation (RC)*, entre outras.

Em Aprendizado de Máquina, geralmente definem-se: o *espaço de entrada*, onde se encontram os objetos a serem analisados, na forma de vetores de *atributos*; o *espaço*

de saída, composto por categorias discretas ou números reais e o *espaço de hipóteses*, que contém os conjuntos de funções e vetores que operam sobre o espaço de entrada e retornam valores para o espaço de saída. Em Learning to Rank, os documentos são costumeiramente representados por conjuntos de *atributos* que refletem a relevância de um documento para aquela consulta. Os valores produzidos de BM25, TF-IDF e Pagerank são atributos típicos. A capacidade de se combinar múltiplos atributos numa única análise é uma grande vantagem de Learning to Rank. [28]

A maior parte dos algoritmos de Learning to Rank pode ser inclusa em uma de três categorias: abordagem por pontos, por pares ou por listas. Na abordagem por pontos, a entrada é o vetor de *atributos* de um único documento, e a saída é a relevância daquele documento. Algoritmos desse estilo incluem o CRR[40], McRank [27] e o Prankster [10]. Na abordagem por pares, a entrada inclui o vetor de atributos de dois documentos, e a saída contém a informação de qual dos dois é mais relevante (valores *{1, -1}*. Exemplos de algoritmos do tipo são Rankboost [18] e RankRLS [32]. Por fim, na abordagem por listas, a entrada contém todo um grupo de documentos associados com uma determinada consulta, e a saída consiste nos graus de relevância de cada documento, ou numa lista ordenada dos documentos. Exemplos são o ListNet [7], AdaRank [51] e o SoftRank [44].

O BRM [12] é um ranker que trabalha unindo diversos rankers menores para melhorar seus resultados, numa forma de boosting semelhante ao que o Rankboost faz. Foram comparados 87 algoritmos [43] de ranking em Learning to Rank sob a ótica de uma métrica chamada *Normalized Winning Number*, que valoriza os algoritmos que, para determinadas base de dados, tiveram desempenho melhor que mais rivais. Foram tidos como melhores, apesar de certas limitações da métrica, os algoritmos ListNet, SmoothRank, FenchelRank, FSMRank and LRUF. Considerando somente a métrica *MAP*, o melhor algoritmo foi o LAC-MR-OR [45].

No contexto de Learning to Rank, a coleção do LETOR [35] foi proposta como benchmark para algoritmos de área. Ela oferece dois datasets: o Gov, composto pelos textos coletados de páginas com domínio .gov nos Estados Unidos em 2003 e 2004. Após poda dos resultados, elas foram divididas em consultas de três naturezas distintas: *topic distillation (TD), homepage finding (HP), e named page finding (NP)*. Há um dataset por tipo de consulta por ano, ou seja, seis datasets. Além disso há o OHSUMED, composto por um compilado de artigos médicos publicados em *journals*, coletados entre 1987 a 1991. Por se tratar de uma coleção de sites, o dataset Gov possui nos atributos de cada documento valores *Query Independent*. Os datasets são, préviamente, divididos em 5 folds e em bases de treino, teste e validação. A benchmark do LETOR é tida como uma das mais completas [43] e robustas da área.

Ainda existem inúmeros problemas em aberto a serem tratados na área [8], principalmente quando se considera o uso dos algoritmos em produção em mecanismos de busca comerciais. Um problema óbvio que se evidencia é o do tempo de execução: apesar do treino poder ser feito offline, o tempo de avaliação de um conjunto de documentos não pode exceder, em situações reais, 50 milisegundos, e muitos dos algoritmos atuais ultrapassam em muito esse valor. Balancear o ganho em performance com a perda de precisão, portanto, é um problema de relevância. Outro problema se revela na robustez de um mecanismo de busca: numa situação real se espera que o algoritmo de ordenação de respostas seja aprimorado constantemente, entretanto é preciso que os resultados apresentados não mudem de forma radical. Mostra-se necessário o desenvolvimento de uma métrica que avalie essa robustez do algoritmo. Uma métrica que se apresenta consiste na medida da probabilidade de um par de vizinhos em rank trocar de lugar num novo resultado de algoritmo. Essa métrica, embora promissora, requer estudos mais aprofundados: a presença de ruído numa base de treino, por exemplo, pode levar à sobreimportância de certos fatores irrelevantes, o que gera pequenas trocas de posição uma reação em cadeia que prejudica seriamente a robustez de um algoritmo.

2.3 Nossa Abordagem

O nosso trabalho se propõe a estudar o potencial que a transformação de atributos tem em Learning to Rank. Essa área, ainda pouco explorada, tem seu potencial exibido quando vemos a melhora que a seleção de atributos trás para algoritmos clássicos de ordenação. A transformação de atributos tem por trás de si uma idéia semelhante à da seleção: assim como ela, a idéia é utilizar as informações que os atributos tem da melhor maneira possível. Ao contrário da seleção, no entanto, não se descarta nenhuma informação por completo. Há potencial de uso, grande ou pequeno, de todas as atributos num processo de transformação, e isso evidencia a redução em perda de informação que podemos ter.

A pergunta que fazemos é: “A transformação de atributos, em Learning to Rank, pode melhorar resultados dos algoritmos de ordenação de respostas clássicos?”. Optamos por usar o autoencoder, uma rede neural que faz tais transformações com excelentes resultados. Fizemos um extenso estudo dos efeitos que a variação nas representações geradas pelos autoencoders tem sobre os resultados dos algoritmos clássicos de ordenação. Descobrimos insights sobre o potencial que o autoencoder tem para tal tarefa, incluindo qual faixa de parâmetros estáticos do mesmo tem o potencial de trazer melhores resultados. Mas, além disso, mostramos o tanto que a transformação de atributos

pode impactar, positivamente, os algoritmos de ordenação de respostas, superando o estado da arte com nossas representações em vários casos.

Capítulo 3

Aprendizado de Representações

Neste trabalho, utilizamos autoencoders para criar representações em dimensionalidade reduzida de uma base de atributos de documentos, sendo estes atributos um subconjunto das diversas métricas de relevância já estabelecidas em Recuperação de Informação. Propõe-se que o uso de tais representações aprimorará os resultados dos algoritmos de ranqueamento tradicionais, uma vez que a representação ideal minimizará ruídos na informação e maximizará a importância das medições mais relevantes, de forma implícita. Para os experimentos e análise dos resultados foi utilizada como teste base a LETOR [35], uma base de dados para benchmarks em Learning to Rank. A criação de modelos tem um alto custo computacional, sendo de grande relevância, portanto, compreender o que influencia o desempenho, na ordenação de respostas, da representação gerada por um autoencoder. A descoberta de um padrão no conjunto de parâmetros bons, ou a simples delimitação do quão fácil ou difícil é encontrar um bom conjunto de parâmetros é de grande relevância caso o autoencoder se apresente como viável para a melhora dos resultados dos desempenhos dos algoritmos.

Mais do que simplesmente buscar novas representações, no entanto, este trabalho estuda o comportamento do autoencoder enquanto gerador de representações para Learning to Rank, além do potencial da transformação de atributos, ou criação de representações, enquanto fator de aprimoramento dos resultados da ordenação de respostas. Para cada base de dados foi feito processamento com uma grande quantidade de autoencoders, cujos hiperparâmetros foram variados uniformemente sobre o espaço total de hiperparâmetros. Assim, diversas novas representações de cada base de dados foram feitas. Tendo como base de qualidade do desempenho da representação gerada por cada autoencoder nos algoritmos de ordenação de respostas, são ainda objetivos deste trabalho: entender o quão fácil ou difícil é gerar representações de qualidade; quais parâmetros estáticos do autoencoder exercem maior influência sobre a qualidade

da representação; buscar propriedades comuns que os autoencoders de qualidade de cada base tenham, de modo a encontrar propriedades em potencial que, para um contexto de Learning to Rank, estejam associadas à melhora de resultados.

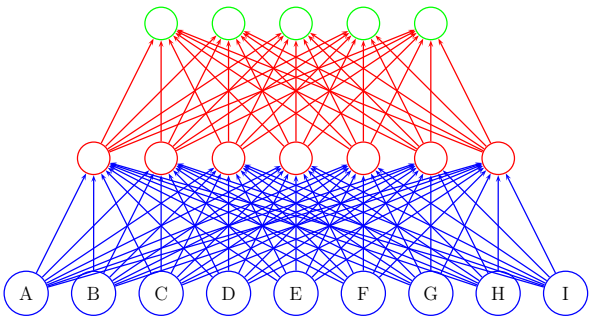

Figura 3.1. Um stacked autoencoder.

Afim de melhor compreendermos os efeitos do autoencoder sobre os algoritmos de Learning to Rank, nós exploramos uma fração de seu espaço de parâmetros estáticos de maneira uniforme, para gerar diversos autoencoders diferentes. Nós exploramos, nesse intuito, os seguintes parâmetros estáticos de stacked denoising autoencoders: quantidade de ruído inserido e número total de nós de cada rede. Ambos têm grande relevância na performance da rede e no tempo de execução de uma série de autoencoders. A maneira como exploramos tal espaço variou conforme a base que estudávamos. Buscamos explorar uma fração razoável do espaço de parâmetros. Se uma base tem, por exemplo, 70 *atributos*, a exploramos de 10 em 10 *atributos*, ou se seja, de 10 para 20 nós, de 20 para 30, e assim por diante. Além disso, optou-se pelo uso de autoencoders piramidais, ou seja, autoencoders que se empilham em camadas progressivamente menores que a camada anterior. Faz-se isso porque espera-se que exista redundância nos atributos das bases de dados: havendo uma correlação entre atributos modelável de forma não linear, há possibilidade de se reduzir a dimensionalidade dos dados sem perder informação, coisa que o autoencoder piramidal, efetivamente, faz.

Capítulo 4

Experimentos

4.1 Abordagem Experimental

Conforme dito anteriormente, a base de dados do LETOR [35] possui dois conjuntos de dados: o OHSUMED, composto por um compilado de artigos médicos publicados em *journals*, coletados entre 1987 a 1991, e o Gov, oriúndo de páginas .gov dos Estados Unidos em 2003 e 2004. O conjunto Gov é dividido em grupos de consultas de três naturezas distintas: *topic distillation (TD), homepage finding (HP), e named page finding (NP)*). Os grupos de consultas TD buscam encontrar uma lista de pontos de entrada para bons websites dedicados a um tópico específico. Os grupos de consultas HP buscam retornar a homepage específica buscada na consulta, e os grupos de consultas NP procuram páginas cujos nomes são idênticos aos da consulta em questão. Para estas duas últimas, HP e NP, falando de forma geral, só existe uma resposta correta para cada consulta. Há, no total, um grupo de consultas de cada tipo (HP, NP e TD) para cada ano (2003 e 2004), ou seja, seis grupos de consulta na base Gov: HP2003, HP2004, NP2003, NP2004, TD2003, TD2004.

Os grupos de consultas são compostos por triplas <*atributos do documento; consulta; taxa de relevância do documento para a consulta*> Todas as triplas foram previamente classificadas por avaliadores humanos como *irrelevantes (0), relevantes (1)*, no caso das bases Gov, e *nada relevantes (0), pouco relevantes (1), muito relevantes (2)* no caso da OHSUMED. Os grupos de consultas são, préviamente, divididos em 5 folds de bases de treino, teste e validação. Cada fold de cada grupo foi processada por 120 configurações diferentes de autoencoder. Os parâmetros variados foram 2: o número de níveis do autoencoder e o número de nós em suas camadas. O parâmetro ruído foi estudado em uma análise mais aprofundada, feita com a base OHSUMED.

Como consequência de nossa exploração uniforme do espaço amostral, houve de-

sigualdade na distribuição de autoencoders por número total de nós. Ou seja, enquanto os autoencoders com número total de nós igual a 40 foram quatro (configurações (40), (30)(10), (20)(20), (20)(10)(10)), os autoencoders com, por exemplo, 120 nós no total, foram dez. A distribuição de quantidade de autoencoders por configuração se vê na figura 4.1.

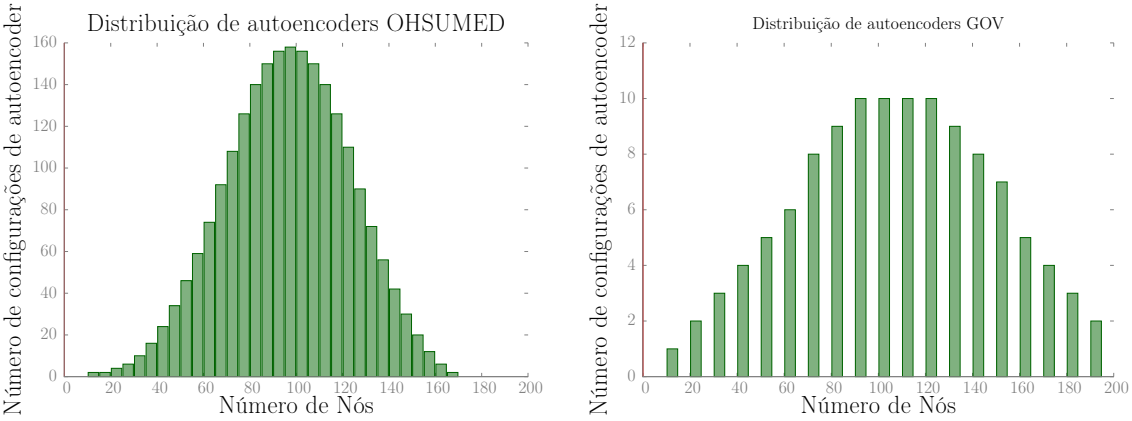

Figura 4.1. Distribuição dos parâmetros de configuração dos autoencoders aplicados à base OHSUMED (1) e idem para a GOV (2). Para a GOV, somente autoencoders de três camadas são exibidos aqui.

A distribuição exibida em 4.1 se repete para os autoencoders com duas camadas. Assim sendo, tivemos 7 autoencoders de uma camada, 28 de duas camadas e 85 de três camadas, totalizando 120 autoencoders, no caso da base Gov. Para os autoencoders aplicados à OHSUMED tal situação se repete. No caso dela, optamos por fazer um experimento menos abrangente e mais denso no espaço de parâmetros. Variamos a quantidade de nós por camada de 5 em 5 e de 20 a 55, uma vez que a base possui 44 atributos. Fixamos também o número de camadas em 3, para estudarmos as representações mais abstratas. Ou seja: nosso menor autoencoder estava parametrizado em (20)(20)(20) e o maior em (55)(55)(55). Além disso, cada autoencoder foi utilizado numa versão sem a aplicação de ruído e outra numa versão com. Assim, tivemos um total de 1525 autoencoders. A distribuição dos mesmos conforme número total de nós é exibida no gráfico 4.1. Para cada base GOV, dados que foram criadadas 120 configurações de autoencoder diferentes, foram geradas 600 representações de dados, 120 para cada uma das 5 folds de cada uma das seis bases de dados, totalizando 3600 representações. Para a base OHSUMED, de forma análoga, tivemos 1525 representações geradas para cada uma das cinco folds, num total de 7625 representações das bases.

A forma de criação dos modelos transformados das bases de dados foi a mesma para todas as configurações de autoencoders estudadas. Nós utilizamos uma abordagem semi-supervisionada: o arquivo de treino, o maior, primeiramente, é utilizado pelo au-

toencoder numa fase de treinamento não supervisionado, onde ele busca simplesmente minimizar o erro de reconstrução da rede. Após isso há uma fase supervisionada, onde um ajuste fino é feito através do arquivo de validação e testes da rede [16, 46].

4.2 Análise dos Resultados

Nós executamos oito algoritmos de ordenação de respostas em Learning to Rank sobre as nossas representações e avaliamos o desempenho das mesmas. Tais algoritmos são utilizados em outras benchmarks da área, como a do próprio LETOR, e estão entre os melhores que existem na literatura. Os algoritmos utilizados são implementados pela biblioteca RankLib [11]. Os algoritmos escolhidos são o Rankboost[18], RankNet[5], Mart, Lambdamart[49], Listnet [7], Adarank [51], Regression L2 e Random Forests [4]. Nós avaliamos o desempenho de cada um deles sob a ótica das métricas MAP e NDCG@10 [23] para todas as representações, ou, transformações, criadas com os autoencoders. O desempenho de todas essas representações foi comparado com o desempenho do algoritmo executado sobre a base inalterada, sendo esta então a nossa baseline.

Os experimentos, isto é, a geração e ordenação de respostas das representações, foram feitos numa máquina com Ubuntu 14.04.2 LTS, Intel(R) Core 7-5820k CPU @ 3.30GHz, GPU Tesla K40c com 12GB de memória e 32GB de memória RAM. O sistema utilizado foi desenvolvido em Python, utilizando-se o Theano [3]. Foram geradas representações para as sete bases de dados do LETOR [35]: OHSUMED, TD2003, NP2003, HP2003, TD2004, NP2004, HP2004. Todas as baselines foram executadas na própria máquina, conforme os parâmetros ditos no site do LETOR. Fora a variação do número de nós de cada camada da rede e do número de neurônios, os outros parâmetros estáticos do autoencoder foram fixados em: pré-treino: taxa de aprendizado 0.01, 15 épocas; finetunning: taxa de aprendizado 0.1, 10000 épocas. A função de ativação foi a sigmóide.

Nas tabelas 4.1 e 4.2, nós apresentamos e analisamos os resultados obtidos: mostramos em histogramas o quão difícil é gerar representações eficazes para as bases de dados em Learning to Rank. Vemos, numa tabela à parte, o quanto que conseguimos melhorar os resultados das baselines e, nos casos onde não houve melhora, o quanto nos aproximamos dela. A tabela evidencia como as transformações dos dados diminuiram a diferença de desempenho entre os diversos algoritmos. Vê-se também que cada algoritmo de ordenação de respostas tem seus resultados aprimorados para bases diferentes, evidenciando que representações melhores podem, também, mitigar as deficiências que

Tabela 4.1. Variação dos melhores resultados obtidos para cada algoritmo em cada base quando comparados com os da baseline, na métrica MAP

	HP2003	*HP2004*	*NP2003*	*NP2004*	*TD2003*	*TD2004*	*OHSUMED*
Regression	0,0502	0,1307	-0,029	0,0621	-0,0107	-0,0257	0,0215
Ranknet	-0,0199	-0,0823	-0,0312	-0,1468	0,0160	0,0065	0,0108
Adarank	-0,0363	0,0125	-0,0575	0,0688	-0,0069	0,0033	0,0086
Rankboost	0,0062	0,0433	-0,0045	0,0631	0,0393	-0,0317	-0,0024
Listnet	-0,0351	0,1553	0,0318	-0,0217	0,0280	0,0132	0,0114
Mart	-0,0265	0,0779	-0,0325	0,0592	0,0510	0,0056	0,0157
Lambdamart	0,0031	0,0046	-0,0046	0,0873	0,0282	-0,0421	0,0067
Random forests	-0,0081	-0,0235	0,0074	0,0035	-0,0142	-0,0435	0,0077

Tabela 4.2. Variação dos melhores resultados obtidos para cada algoritmo em cada base quando comparados com os da baseline, na métrica NDCG@10

	HP2003	*HP2004*	*NP2003*	*NP2004*	*TD2003*	*TD2004*	*OHSUMED*
Regression	0,0582	0,141	-0,02	0,0495	-0,0136	-0,0224	0,0298
Ranknet	-0,0184	-0,1495	-0,0185	-0,0232	0,0027	0,0198	0,02
Adarank	-0,0172	0,045	-0,0056	0,0733	0,0276	0,0248	0,0078
Rankboost	0,0069	0,0297	0,006	0,0346	0,0454	-0,0335	0,0224
Listnet	0,0363	-0,1069	-0,014	0,1351	0,0282	0,0095	0,0184
Mart	-0,0297	0,0858	-0,0182	0,0483	0,0529	0,0058	0,0196
Lambdamart	0,0054	0,0657	-0,0075	0,1583	-0,0045	0,0058	0,0185
Random forests	-0,002	0,0437	-0,0287	0,0184	-0,0301	-0,047	0,0105

esses algoritmos possam vir a ter.

Fazemos ainda uma análise da influência que o parâmetro estático *número total de nós do autoencoder* tem sobre o desempenho das representações, com dados apresentados tanto nos histogramas quanto em tabelas numéricas. Para esta análise, verificamos a influência que tal parâmetro exerce sobre os resultados de cada algoritmo, de cada base de dados e de cada métrica de avaliação separadamente. Os resultados são visualizáveis de duas formas: (1) através dos histogramas, onde a altura das barras representa a quantidade de autoencoders em uma determinada faixa de escore (métrica MAP ou NDCG@10) e a cor das barras representa o quanto cada faixa de parâmetros (número total de nós) prevaleceu em cada faixa de escore; (2) uma tabela, onde vemos qual é a distribuição esperada de autoencoders para cada faixa de parâmetros (0 a 30 nós, 30 a 60, e assim por diante) e como cada faixa se saiu na prática. Ambas as visualizações mostram, no fim das contas, se há alguma faixa de número total de nós

que tende a ter resultados melhores que o esperado.

Abaixo, as análises de desempenho foram subdivididas por métrica, base, algoritmo e análise geral. Para cada caso, nós agregamos todos os resultados relativos ao objeto em análise e buscamos ver se houve prevalência de algum conjunto de configurações entre os melhores resultados, em relação a uma distribuição uniforme dos resultados, conforme a quantidade de autoencoders em cada faixa de parâmetros (ver Figura 4.1). Para a métrica MAP, por exemplo, todas as bases GOV e algoritmos executados na métrica MAP foram agregados, e a análise tem por objetivo mostrar se o parâmetro analisado se destaca entre os melhores independente de qual algoritmo ou base GOV foi usada, na média. Como a base de dados OHSUMED é, além de estudada com um número muito maior de autoencoders, diferente da GOV em seus atributos e número de atributos, as análises agregadas não a incluíram. Suas análises são vistas à parte. A base GOV, ressalte-se, possui 64 atributos, enquanto as bases OHSUMED possuem 44.

4.3 Análise por métrica

Nesta sessão nós vemos se há alguma faixa de parâmetros, isto é, de número total de nós, tende a ter mais resultados bons do que se esperaria numa distribuição uniforme. Fazemos a mesma análise, contemplando a base OHSUMED, para duas taxas de ruído: 0% e 30%. Uma taxa de ruído de 30% indica que 30% dos dados da entrada serão corrompidos antes do treino da rede.

4.3.1 MAP

Vemos na Tabela 4.3 que a faixa de nós totais de 120 a 180 teve sutil prevalência entre os melhores resultados, sendo mais frequentes ali do que se esperaria de um resultado distribuído uniformemente. Tal resultado agrega todos os algoritmos executados sobre todas as bases GOV, sendo, portanto, bastante robusto: quanto mais resultados agregados, mais relevante é o que eles mostram. A taxa de corrupção, por sua vez, não apresentou diferenciais relevantes na busca por melhores resultados.

Tabela 4.3. Desempenho dos autoencoders por número total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.4%	2.26%	2.64%
30.0 a 60.0	9.87%	8.53%	12.24%
60.0 a 90.0	20.22%	18.7%	22.91%
90.0 a 120.0	25.26%	25.24%	25.31%
120.0 a 150.0	22.78%	23.56%	21.39%
150.0 a 180.0	13.54%	14.75%	11.4%
180.0 a 210.0	5.08%	5.99%	3.48%
210.0 a 240.0	0.85%	0.97%	0.64%
No. Combinações Estudadas	5664	3621	2043

Tabela 4.4. Desempenho dos autoencoders por nível de corrupção, contemplando somente a base OHSUMED.

Nível de ruído nos dados	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0	50.01%	49.76%	54.31%
0.3	49.99%	50.24%	45.69%
No. Combinações Estudadas	12537	11841	696

4.3.2 NDCG@10

Na Tabela 4.5 é claro que houve prevalência na faixa de resultados de 120 a 180 nós, havendo indicação clara que os autoencoders com essas características tenderam a dar resultados melhores que os demais. A taxa de corrupção não apresentou diferenciais relevantes na busca por melhores resultados.

Tabela 4.5. Desempenho dos autoencoders por numero total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para algum algoritmo, considerando a metrica NDCG@10, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.4%	2.22%	2.89%
30.0 a 60.0	9.85%	8.62%	13.21%
60.0 a 90.0	20.14%	18.95%	23.41%
90.0 a 120.0	25.34%	25.27%	25.51%
120.0 a 150.0	22.81%	23.51%	20.91%
150.0 a 180.0	13.52%	14.82%	9.99%
180.0 a 210.0	5.08%	5.72%	3.35%
210.0 a 240.0	0.85%	0.89%	0.72%
No. Combinações Estudadas	5664	4143	1521

Tabela 4.6. Desempenho dos autoencoders por nível de corrupção, contemplando somente a base OHSUMED. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para algum algoritmo na métrica NDCG@10, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

Nível de ruído nos dados	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0	50.02%	49.63%	51.14%
0.3	49.98%	50.37%	48.86%
No. Combinações Estudadas	12537	9301	3236

4.3.3 Sumário

É possível ver, conforme os resultados mostram, que os resultados não exibem variação relevante de padrão por métrica. Ambas favoreceram as faixas de 120 a 180 nós totais, os resultados dessas faixas estando presentes aí em 1 ponto percentual acima da distribuição base. Isso são significa que essa faixa seja boa em qualquer caso em Learning to Rank, isto resta ver na análise mais aprofundada, feita abaixo.

Para ambas as métricas a taxa de corrupção não se mostrou como diferencial relevante na busca por melhores resultados. O que vemos, em suma, é que como não há variação nos padrões encontrados por métrica, a métrica utilizada não influencia no desempenho que uma faixa de nós totais tem sobre os resultados dos algoritmos.

4.4 Análise por base de dados

Vimos nas Tabelas 4.3 e 4.5 que a faixa de 30 a 90 nós de autoencoders apresentou os melhores resultados, considerando o agregado de todos os algoritmos executados em todas as bases GOV para tais métricas. Abaixo, buscamos ver se o padrão se repete quanto analisamos os resultados de cada base de dados separadamente. Ou seja: uma vez que vemos que os resultados de cada média tem clara tendência, buscamos ver se essa tendência é geral ou se limita a somente parte dos resultados.

4.4.1 Gov - HP

Na base HP, busca-se por uma URL específica em meio às páginas da base de dados: a URL da homepage de uma página buscada na consulta. A tabela 4.7 mostra uma sutil prevalência de resultados bons na mesma faixa exibida na análise por métrica, a de 120 a 180 nós totais. Tal prevalência segue da linha da que se viu nos resultados exibidos por métrica: nessa faixa, os autoencoders estiveram presentes entre os melhores resultados um ponto percentual a mais do que se esperaria numa distribuição uniforme.

Tabela 4.7. Desempenho dos autoencoders por número total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para algum algoritmo sobre a base HP, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.38%	2.26%	3.42%
30.0 a 60.0	9.8%	8.52%	20.29%
60.0 a 90.0	20.26%	19.1%	29.83%
90.0 a 120.0	25.29%	25.66%	22.25%
120.0 a 150.0	22.78%	23.61%	15.89%
150.0 a 180.0	13.56%	14.46%	6.11%
180.0 a 210.0	5.08%	5.49%	1.71%
210.0 a 240.0	0.85%	0.89%	0.49%
No. Combinações Estudadas	3776	3367	409

4.4.2 Gov - NP

Na base NP, busca-se pela página cujo título é semelhante ou igual ao buscado. Temos aqui resultamos semelhantes aos da base HP, em faixa prevalente e força: a faixa de 120 a 180 nós esteve presente entre os melhores resultados com uma frequência de 1 ponto percentual acima do que seria esperado. Ou seja: apesar de somente 13.5% dos autoencoders estarem na faixa de 150 a 180 nós, eles representaram 15.1% dos resultados bons.

Tabela 4.8. Desempenho dos autoencoders por número total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para algum algoritmo sobre a base NP, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.38%	2.21%	3.47%
30.0 a 60.0	9.9%	8.32%	19.85%
60.0 a 90.0	20.02%	18.02%	32.56%
90.0 a 120.0	25.34%	25.42%	24.86%
120.0 a 150.0	22.85%	24.1%	15.03%
150.0 a 180.0	13.56%	15.11%	3.85%
180.0 a 210.0	5.08%	5.83%	0.39%
210.0 a 240.0	0.85%	0.98%	0.00%
No. Combinações Estudadas	3776	3257	519

4.4.3 Gov - TD

Aqui o padrão visto até então se altera. Há prevalência sutil entre os autoencoders que possuem 150 a 210 nós. Aqui é importante ressaltar que a base de dados TD é substancialmente mais difícil de se analisar do que as demais: vemos isso nos resultados de suas baselines que tem suas pontuações em MAP e NDCG@10 muito abaixo das das bases HP e NP, ambas com resultados muito mais próximos. A base TD busca encontrar bons pontos de entrada para um determinado tema buscado na consulta, algo, é possível argumentar, muito mais subjetivo do que encontrar uma URL buscada ou um site cujo título é o que se buscou. Para autoencoders, essa diferença nos resultados indica que a melhora que novas representações oferecem estão condicionadas à natureza da base estudada, ou seja, suas características internas.

Tabela 4.9. Desempenho dos autoencoders por número total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para algum algoritmo na base TD, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.44%	2.28%	2.5%
30.0 a 60.0	9.88%	9.47%	10.05%
60.0 a 90.0	20.26%	20.35%	20.22%
90.0 a 120.0	25.26%	23.6%	25.99%
120.0 a 150.0	22.75%	21.67%	23.22%
150.0 a 180.0	13.48%	14.82%	12.9%
180.0 a 210.0	5.08%	6.93%	4.29%
210.0 a 240.0	0.85%	0.88%	0.83%
No. Combinações Estudadas	3776	1140	2636

4.4.4 Ohsumed

Vemos que a base OHSUMED, por ter sido mais estudada, possui resultados muito mais robustos que os apresentados nas bases GOV. Aqui, os resultados dos autoencoders na faixa de 90 a 120 nós apresentam somente uma leve melhora em relação à distribuição uniforme, na casa de um ponto percentuais. Os resultados da faixa de 120 a 150 nós apresentam uma melhora mais relevante, na casa de 3 pontos percentuais. Ressalte-se que, tanto por ter uma quantidade menor de atributos que as bases GOV quanto por ter uma natureza de dados diferente (os textos não são websites, mas artigos médicos), as características da base OHSUMED diferem muito das outras e, entretanto, ela segue a tendência percebida nas análises gerais da base GOV, com foco nas métricas.

Tabela 4.10. Desempenho dos autoencoders por número total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para algum algoritmo sobre a base OHSUMED, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu numero total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	0.89%	0.85%	0.91%
30.0 a 60.0	12.11%	10.75%	12.73%
60.0 a 90.0	39.85%	35.46%	41.87%
90.0 a 120.0	37.38%	38.47%	36.88%
120.0 a 150.0	9.44%	13.48%	7.59%
150.0 a 180.0	0.32%	0.98%	0.02%
No. Combinações Estudadas	25074	7876	17198

4.4.5 Sumário

Vemos aqui que, enquanto as bases HP e NP apresentaram resultados semelhantes, com leve favorecimento das faixas de parâmetros de 120 a 180 nós totais, a base OHSUMED apresentou um favorecimento semelhante na faixa de 90 a 120 nós totais e mais robusto na faixa de 120 a 150 nós. A base TD mostrou favorecimento na faixa de 150 a 210 nós. Isso mostra que a base de dados analisada, por mais que possua atributos iguais (no caso de HP, NP e TD) influencia de formas diferentes o desempenho que determinada faixa de nós tem sobre os resultados dos algoritmos. A base de dados utilizada é, portanto, um fator de alguma relevância na hora de se determinar quais autoencoders utilizar para aprimorar seus resultados.

4.5 Análise por algoritmo

Abaixo, analisamos os resultados agregados por algoritmo, independente de métrica ou base de dados GOV utilizada, e mostramos os gráficos de cada resultado, separados por métrica e base de dados. Nos referimos constantemente à Tabela 4.1 para comparar o desempenho de cada algoritmo com as baselines. Mais do que analisar os autoencoders de acordo com seu número total de nós ou taxa de corrupção, aqui nós buscamos ver como o resultado de cada algoritmo melhorou, ou não, com auxílio dos autoencoders.

Temos, portanto, dois tipos de informação sendo analisadas: (1) os autoencoders de qual faixa de nós/taxa de corrupção ofereceram os melhores resultados para esse algoritmo? (2) tais resultados foram suficientes para superar a baseline, isto é, a execução dos algoritmos sobre a base de dados inalterada, ou os autoencoders conseguem se aproximar do resultado base, mas sempre piorando-o?

As imagens 4.2 a 4.17 agregam os histogramas que mostram como cada grupo de autoencoders se comportou pra cada conjunto de dados, métrica e algoritmo. O eixo x de cada gráfico representa as faixas de desempenho do algoritmo em questão para a métrica em questão. O eixo y mostra a quantidade de autoencoders em uma determinada faixa desempenho. As cores de cada barra separam os grupos de autoencoders, de acordo com seu número total de nós. A linha vermelha de cada gráfico representa o desempenho da baseline.

4.5.1 Adarank

O Adarank é um algoritmo de boosting [51] que cria repetidamente 'ordenadores fracos' e os combina linearmente para fazer predições. Parte do seu processo de treino envolve

a minimização de uma função de perda exponencial que foi definida diretamente sobre as métricas MAP e NDCG. Em seus gráficos podemos analisar o efeito das representações criadas pelos autoencoders sobre ele. A Tabela 4.1 e os gráficos mostram que as representações resultaram em ganhos de desempenho pouco expressivo para quase todas as bases. As bases HP2003 e NP2003 tiveram perda expressiva de desempenho, da ordem de 4 a 6 pontos percentuais. Na base NP2004 obteve-se um ganho expressivo de desemepenho, de quase 7 pontos percentuais sobre a baseline.

Tabela 4.11. Desempenho dos autoencoders por número total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo Adarank, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuicao dos autoencoders de acordo com seu número total de nós, ou seja: 10,1% dos autoencoders que analisaram a base GOV tem entre 30 e 60 nós, no total. A tabela mostra, no entanto, que 15% dos autoencoders no grupo dos piores tinham entre 30 e 60 nós, enquanto somente 8,23% dos mlehores autoencoders estavam nessa faixa de nós. Ou seja, os autoencoders desta faixa de nós totais tenderam a dar resultados piores que a média.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0,0 a 30,0	2,26%	1,59%	4,32%
30,0 a 60,0	10,1%	8,23%	15,85%
60,0 a 90,0	20,2%	19,55%	22,19%
90,0 a 120,0	25,21%	24,04%	28,82%
120,0 a 150,0	22,81%	23,67%	20,17%
150,0 a 180,0	13,49%	15,15%	8,36%
180,0 a 210,0	5,08%	6,64%	0,29%
210,0 a 240,0	0,85%	1,12%	0,00%
No. Combinações Estudadas	1416	1069	347

É possível observar nos gráficos da Figura 4.2 que há grande variação no desempenho que as novas representações proporcionaram. O desempenho das representações varia de quase nulo a moderadamente inferior à baseline, com conjuntos de cinco representações ou menos obtendo desempenho próximo à baseline ou superior à mesma. Na métrica NDCG, com resultados apresentados na Figura 4.3, a variação é muito menor, mas o desempenho segue o mesmo padrão.

A Tabela 4.11 mostra que, agregando todos esses resultados, a faixa de nós totais de 120 a 210 nós apresentaram melhoras, com cerca de 1-1.5 ponto percentual de

Tabela 4.12. Desempenho dos autoencoders por nível de corrupção, contemplando somente a base OHSUMED. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo adarank, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

Nível de ruído nos dados	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0	50.0%	46.09%	68.41%
0.3	50.0%	53.91%	31.59%
No. Combinações Estudadas	2948	2432	516

aumento, em todos os casos. Vemos, assim, que os resultados para o algoritmo Adarank, como um todo, seguiram o padrão de favorecerem a faixa de nós entre 120 a 180, estendendo essa faixa para a faixa de 180 a 210 nós. O aprimoramento em relação à baseline foi expressivo somente para a base NP2004, e nas restantes a variação obtida foi irrelevante. No caso das taxas de corrupção, 53.9% dos resultados bons possuiam taxa de corrupção 0.3, contra um percentual de resultados bons esperado de 50%.

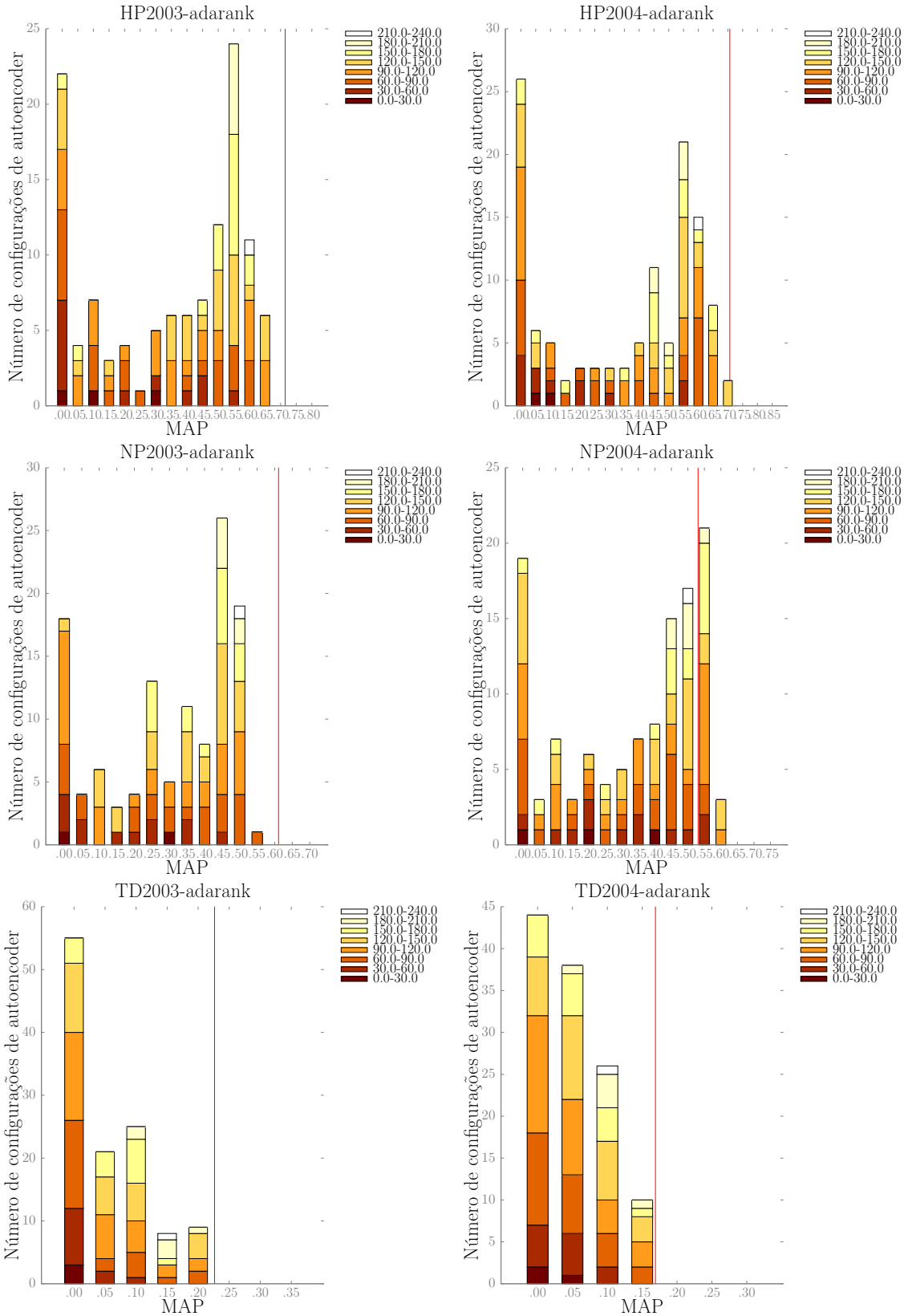

Figura 4.2. Adarank - Desempenho para a métrica MAP na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós. A linha vermelha representa a baseline para cada caso.

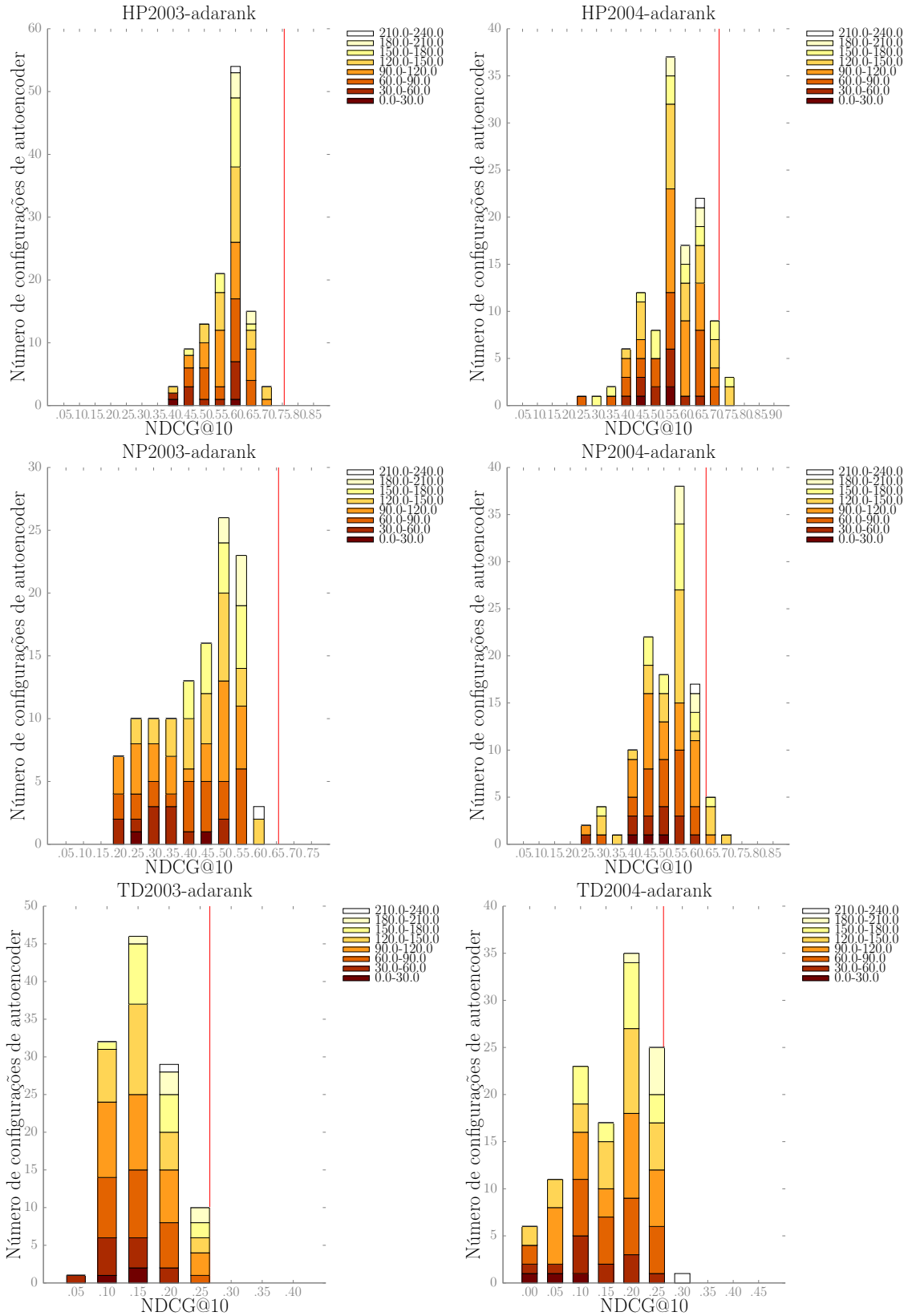

Figura 4.3. Adarank - Desempenho para a métrica NDCG@10 na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós. A linha vermelha representa a baseline para cada caso.

4.5.2 Rankboost

O Rankboost [18] é um algoritmo que, de forma similar ao AdaRank, combina diversos ordenadores mais fracos para obter um bom resultado. A Tabela 4.1 mostra que o Rankboost teve ganho expressivo de desempenho para as bases HP2004, NP2004, TD2003, na faixa de 4 a 6 pontos percentuais. Para as demais bases, teve-se uma variação de desempenho irrelevante.

Tabela 4.13. Desempenho dos autoencoders por número total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo Rankboost, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.12%	1.31%	4.58%
30.0 a 60.0	10.17%	9.47%	12.32%
60.0 a 90.0	20.34%	20.71%	19.2%
90.0 a 120.0	25.42%	25.87%	24.07%
120.0 a 150.0	22.53%	22.02%	24.07%
150.0 a 180.0	13.49%	13.87%	12.32%
180.0 a 210.0	5.08%	5.81%	2.87%
210.0 a 240.0	0.85%	0.94%	0.57%
No. Combinações Estudadas	1416	1067	349

Tabela 4.14. Desempenho dos autoencoders por nível de corrupção, contemplando somente a base OHSUMED. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo rankboost, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

Nível de ruído nos dados	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0	50.0%	48.85%	69.05%
0.3	50.0%	51.15%	30.95%
No. Combinações Estudadas	2948	2780	168

Vemos, ao contrário do que ocorreu com o algoritmo Adarank, que os resultados para o Rankboost variaram bem menos em desempenho. Entretanto, da mesma forma

que aquele, os autoencoders que apresentaram resultados melhores que as baselines foram pouco numerosos em todos os casos.

A Tabela 4.13 mostra que nenhuma taxa de nós total prevaleceu entre os melhores resultados. Em relação às taxas de corrupção, a taxa de 0.3 ficou 1.15% ponto percentual acima do esperado, um indicativo fraco de superioridade de tal taxa sobre a outra.

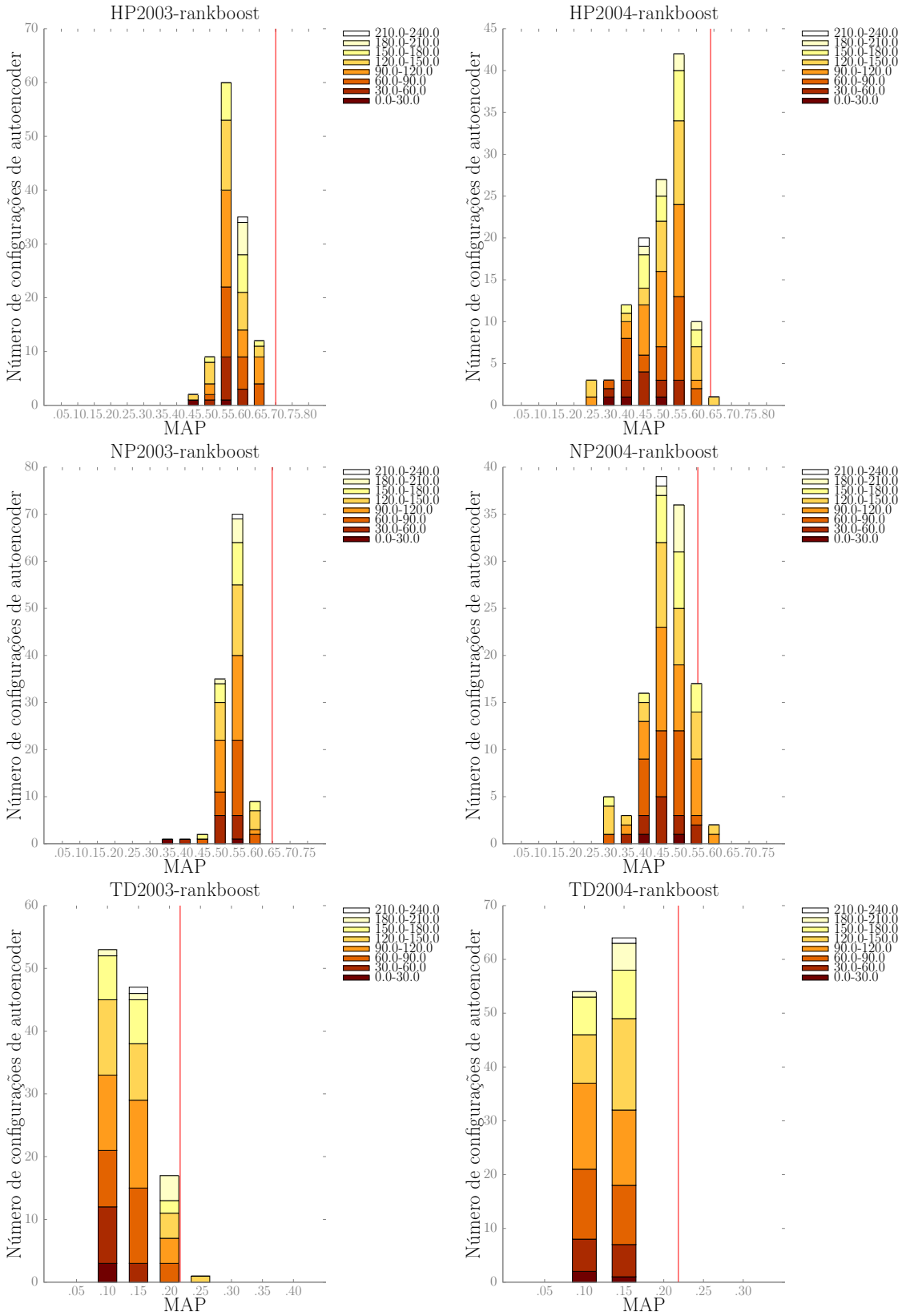

Figura 4.4. Rankboost - Desempenho para a métrica MAP na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós. A linha vermelha representa a baseline.

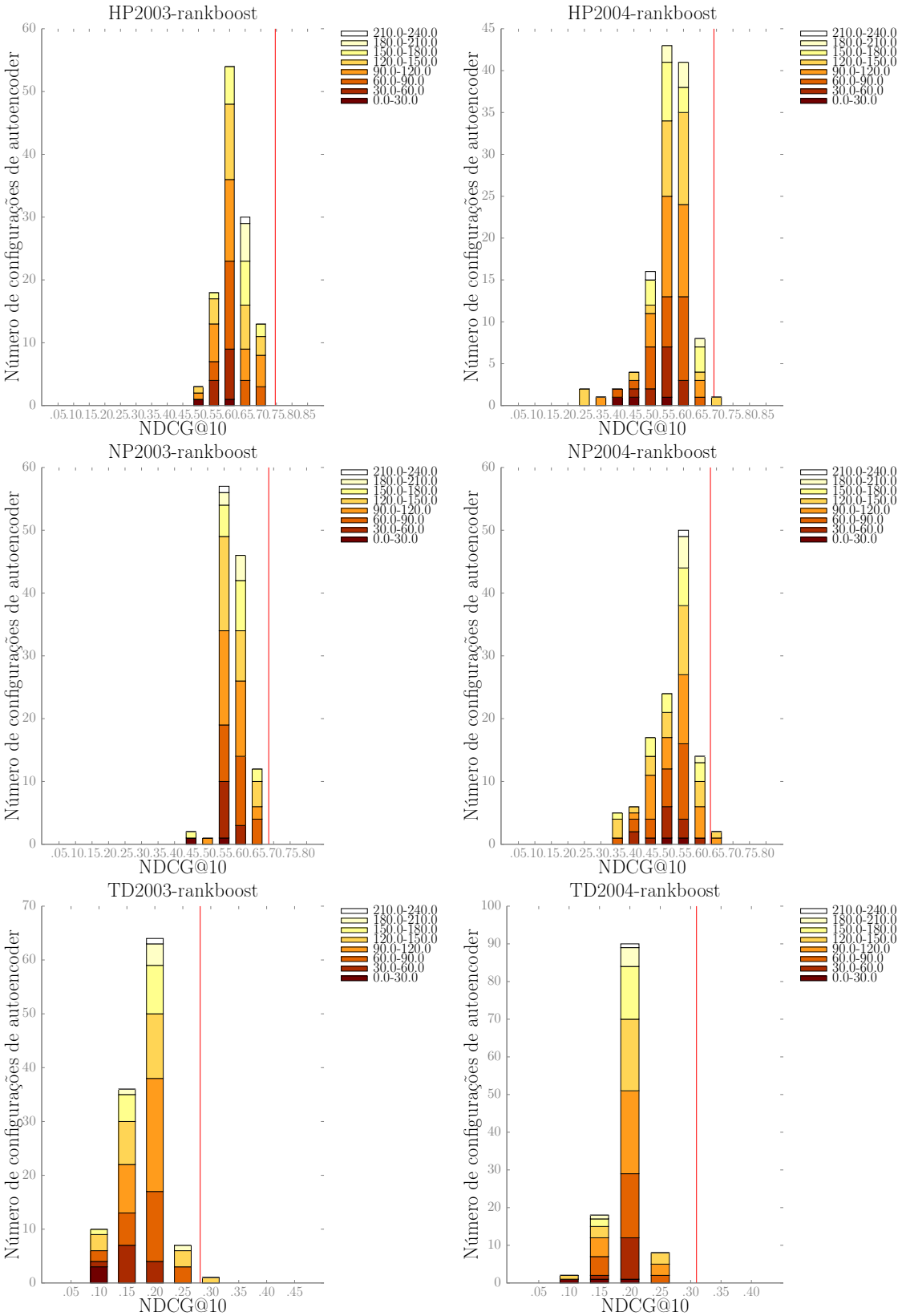

Figura 4.5. Rankboost - Desempenho para a métrica NDCG@10 na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós. A linha vermelha representa a baseline.

4.5.3 Listnet

O Listnet é um algoritmo que classifica os dados utilizando um modelo probabilistico de função de perda numa abordagem por listas, que busca classificar os documentos dividindo-os por grupos, de acordo com suas consultas, e juntando os resultados. As bases HP2003 e NP2004 exibiram perda de desempenho, na faixa de 2 a 3 pontos percentuais; A base NP2003 e TD2003 tiveram leve aumento de desempenho, na faixa de 1.5 a 3 pontos percentuais. A base HP2004 teve uma subida muito grande de desempenho, subindo em 15.5 pontos percentuais.

Tabela 4.15. Desempenho dos autoencoders por número total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo Listnet, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.54%	2.6%	2.43%
30.0 a 60.0	9.96%	8.93%	12.14%
60.0 a 90.0	20.13%	17.76%	25.17%
90.0 a 120.0	25.14%	24.4%	26.71%
120.0 a 150.0	22.74%	23.99%	20.09%
150.0 a 180.0	13.56%	15.37%	9.71%
180.0 a 210.0	5.08%	6.02%	3.09%
210.0 a 240.0	0.85%	0.93%	0.66%
No. Combinações Estudadas	1416	963	453

Os gráficos, das Figuras 4.6 e 4.7, exibem grande variação de resultados para a bases HP2004 e menos nas demais. Novamente, poucos são os resultados que superam a baseline, sendo o base HP2004 do MAP e a NP2004 do NDCG os grandes destaques.

Aqui novamente vemos o destaque sutil entre as melhores representações nas faixas de 120 a 210 nós totais. Nenhuma taxa de corrupção prevaleceu entre os melhores resultados.

Tabela 4.16. Desempenho dos autoencoders por nível de corrupção, contemplando somente a base OHSUMED. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo listnet, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

Nível de ruído nos dados	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0	50.0%	49.97%	50.03%
0.3	50.0%	50.03%	49.97%
No. Combinações Estudadas	2948	1475	1473

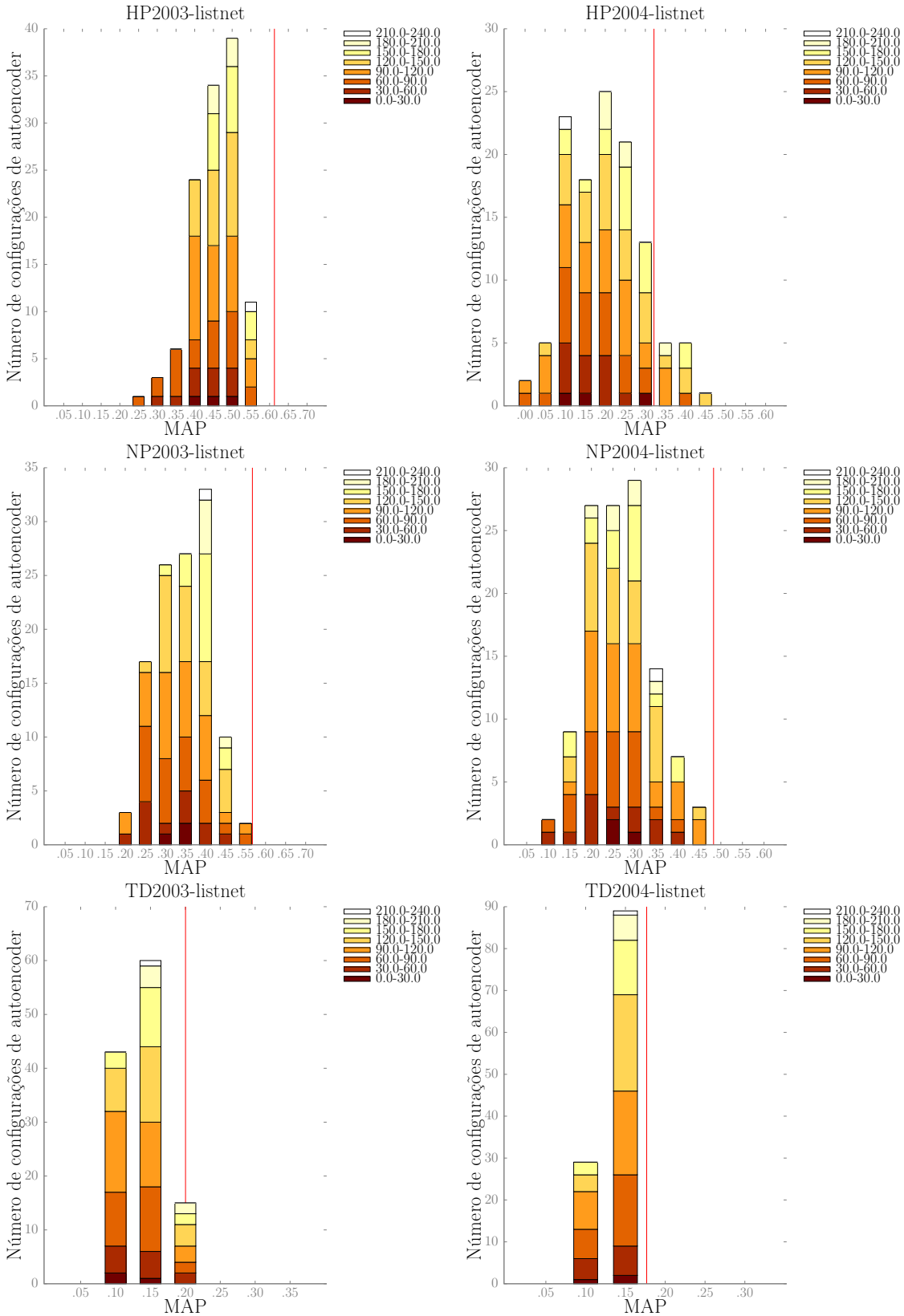

Figura 4.6. Listnet - Desempenho para a métrica MAP na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós. A lista vermelha representa a baseline.

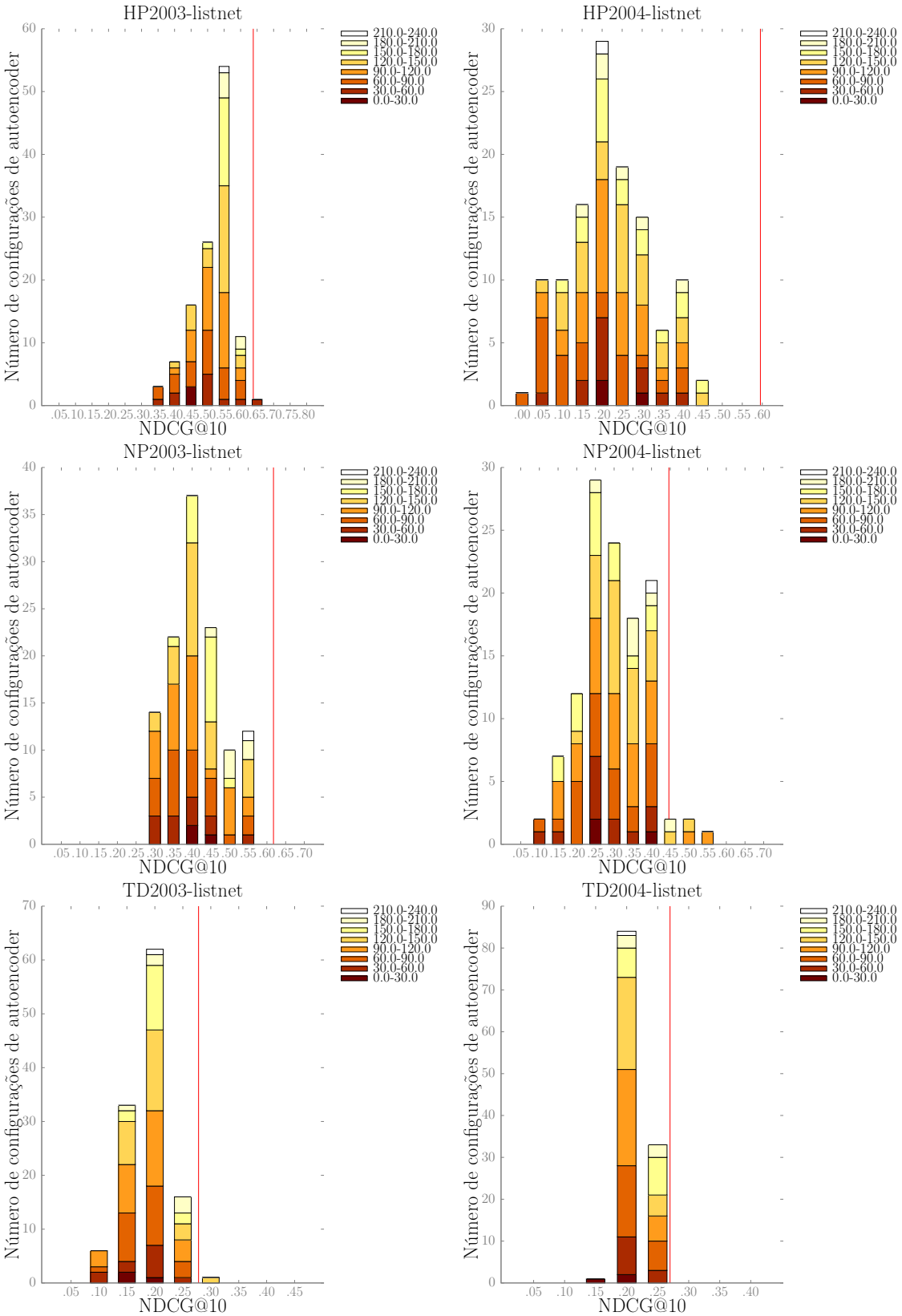

Figura 4.7. Listnet - Desempenho para a métrica NDCG@10 na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós. A lista vermelha representa a baseline.

4.5.4 Ranknet

O algoritmo Ranknet aplica, par a par nas entradas da base de dados, uma rede neuronal de ordenação de respostas associada a uma função de custo probabilistica para treiná-la. Aqui, vemos perdas grandes de desempenho, em relação à baseline, nas bases HP e NP, sem mudança expressiva de desempenho nas outras bases.

Tabela 4.17. Desempenho dos autoencoders por número total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo Ranknet, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.47%	2.52%	2.38%
30.0 a 60.0	9.96%	9.66%	10.5%
60.0 a 90.0	20.2%	18.22%	23.76%
90.0 a 120.0	25.07%	24.7%	25.74%
120.0 a 150.0	22.88%	23.6%	21.58%
150.0 a 180.0	13.49%	14.82%	11.09%
180.0 a 210.0	5.08%	5.6%	4.16%
210.0 a 240.0	0.85%	0.88%	0.79%
No. Combinações Estudadas	1416	911	505

Tabela 4.18. Desempenho dos autoencoders por nível de corrupção, contemplando somente a base OHSUMED. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo ranknet, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

Nível de ruído nos dados	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0	50.0%	49.8%	50.2%
0.3	50.0%	50.2%	49.8%
No. Combinações Estudadas	2948	1480	1468

Novamente vemos aqui uma variação clara nos resultados gerados pelo autoencoder, para ambas as métricas, sem padrão claro visível. Dentre as configurações de

melhor desempenho, houve leve destaque para as configurações de 120 a 180 nós, somente. Nenhuma taxa de corrupção prevaleceu entre os melhores resultados aqui.

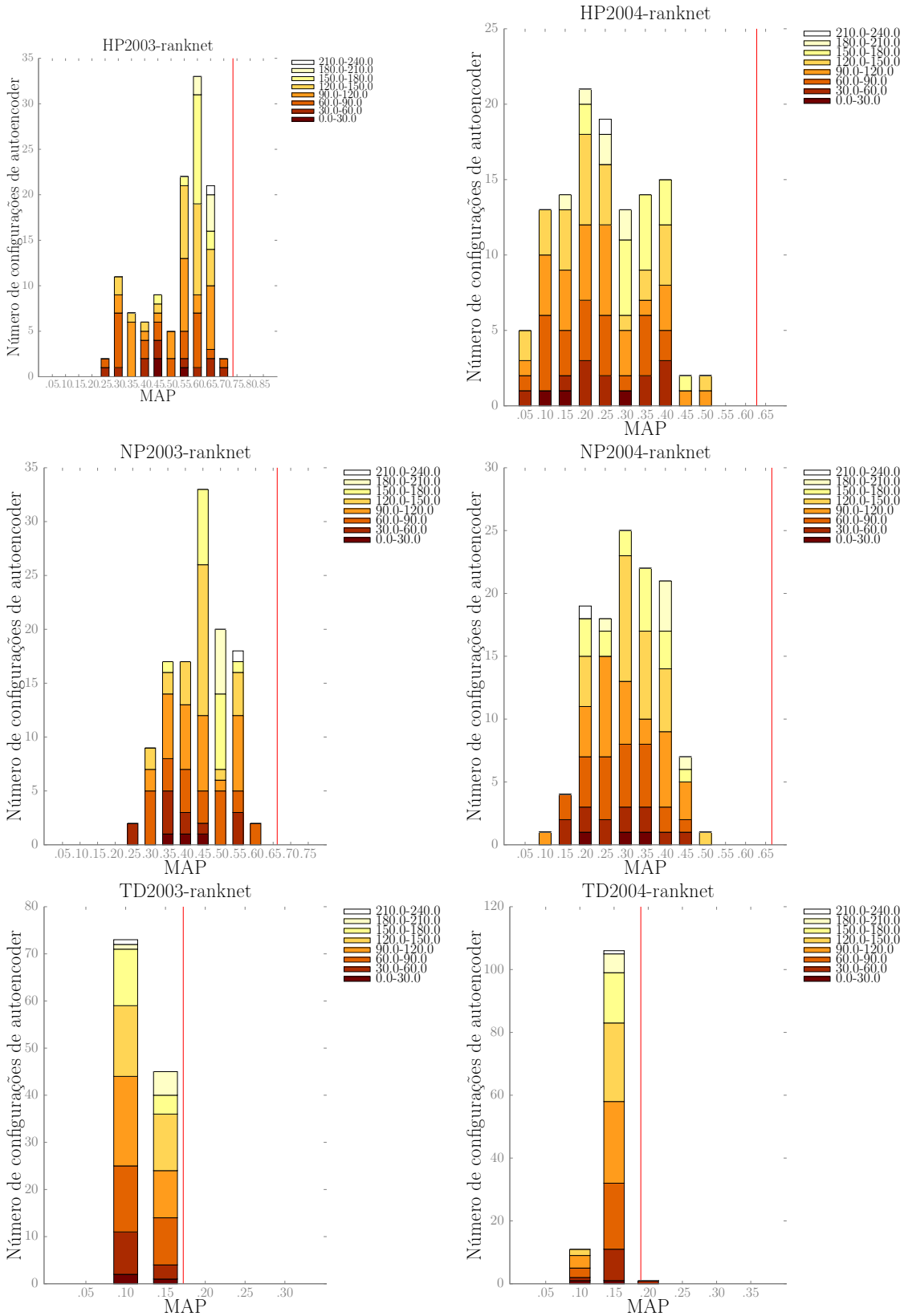

Figura 4.8. Ranknet - Desempenho para a metrica MAP na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós.

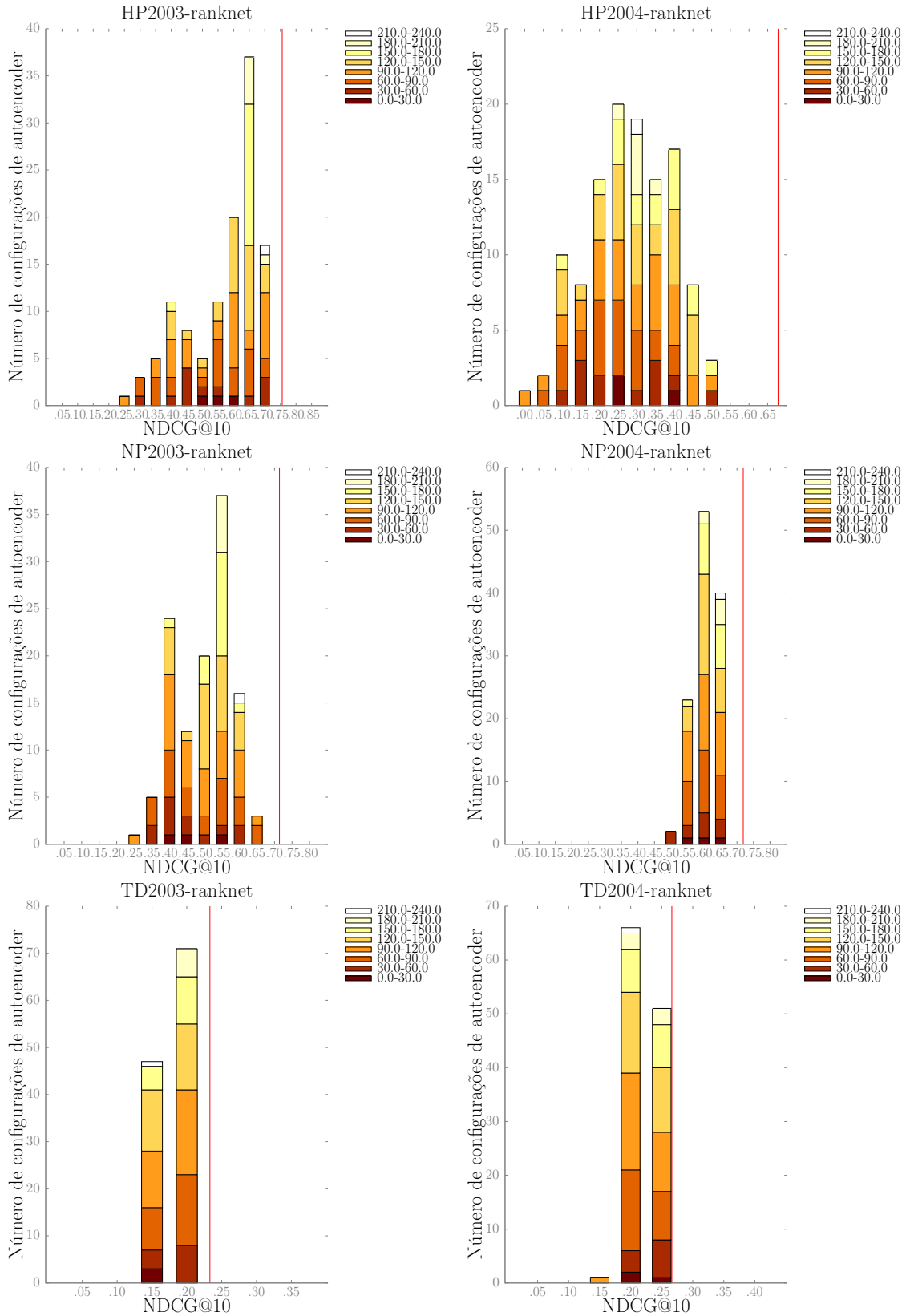

Figura 4.9. Ranknet - Desempenho para a métrica NDCG@10 na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós.

4.5.5 Regression

A regressão é um dos algoritmos mais clássicos e simples que existem para tarefas de aprendizado de máquina. Vemos na Tabela 4.1 que o algoritmo obtém grandes ganhos de desempenho para as bases HP e NP2004, com as bases NP2003 e TD variando negativamente em menor escala e a OHSUMED variando positivamente, também em menor escala.

Tabela 4.19. Desempenho dos autoencoders por número total de nos. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo Regression L2, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nos.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.47%	2.36%	3.45%
30.0 a 60.0	10.1%	9.44%	15.86%
60.0 a 90.0	19.77%	20.54%	13.1%
90.0 a 120.0	25.42%	25.65%	23.45%
120.0 a 150.0	22.74%	22.27%	26.9%
150.0 a 180.0	13.56%	13.45%	14.48%
180.0 a 210.0	5.08%	5.51%	1.38%
210.0 a 240.0	0.85%	0.79%	1.38%
No. Combinações Estudadas	1416	1271	145

Tabela 4.20. Desempenho dos autoencoders por nível de corrupção, contemplando somente a base OHSUMED. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo regression, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

Nível de ruído nos dados	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0	50.0%	51.54%	29.32%
0.3	50.0%	48.46%	70.68%
No. Combinações Estudadas	4438	4131	307

O desempenho das representações seguiu conforme a distribuição esperada, como

evidencia a Tabela 4.19. A taxa de corrupção de 0.0 prevaleceu levemente sobre a taxa de 0.3.

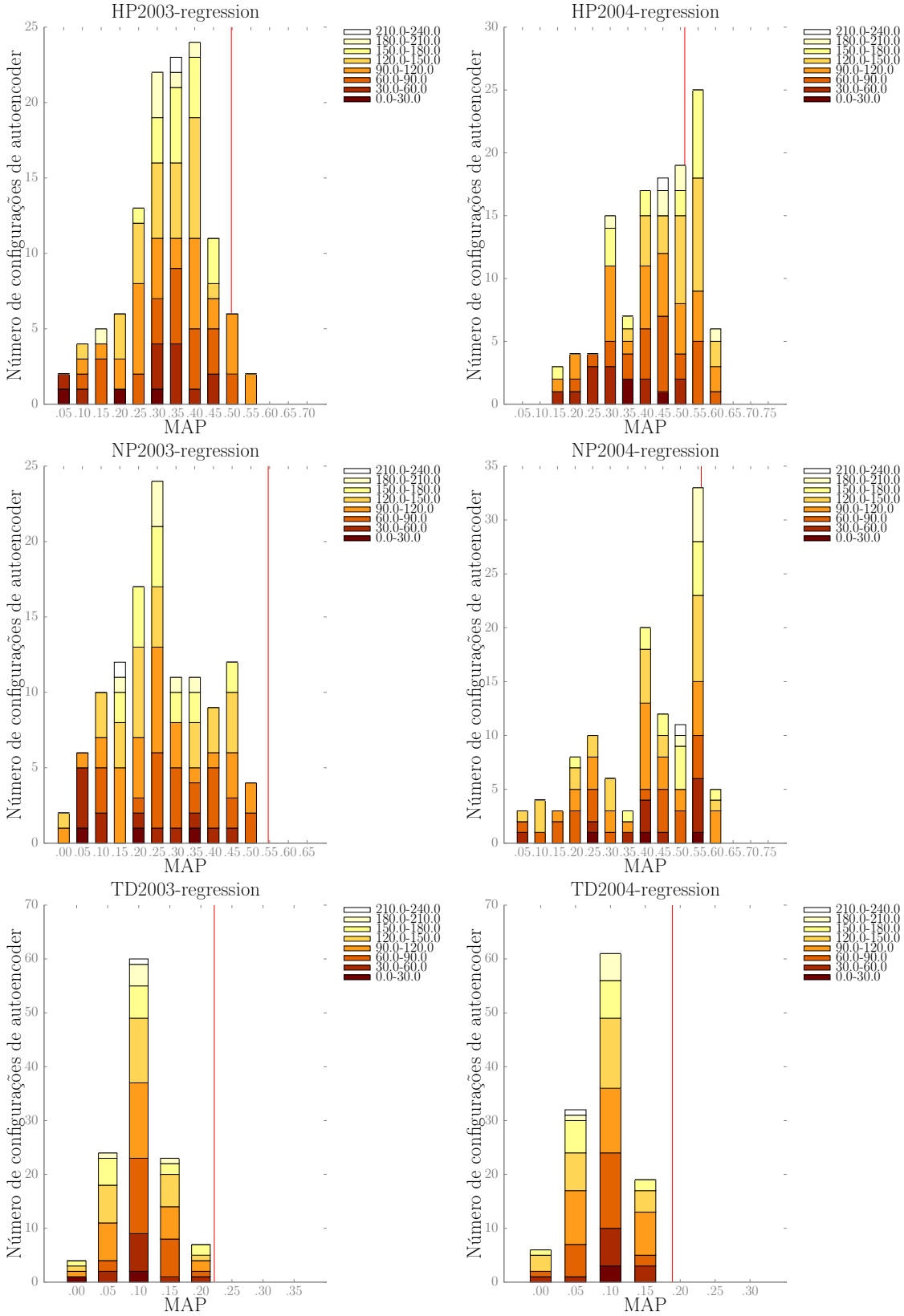

Figura 4.10. Regression - Desempenho na base GOV para a métrica MAP. As cores das barras indicam a faixa de parametros daquele autoencoder, numa divisão por número total de nós.

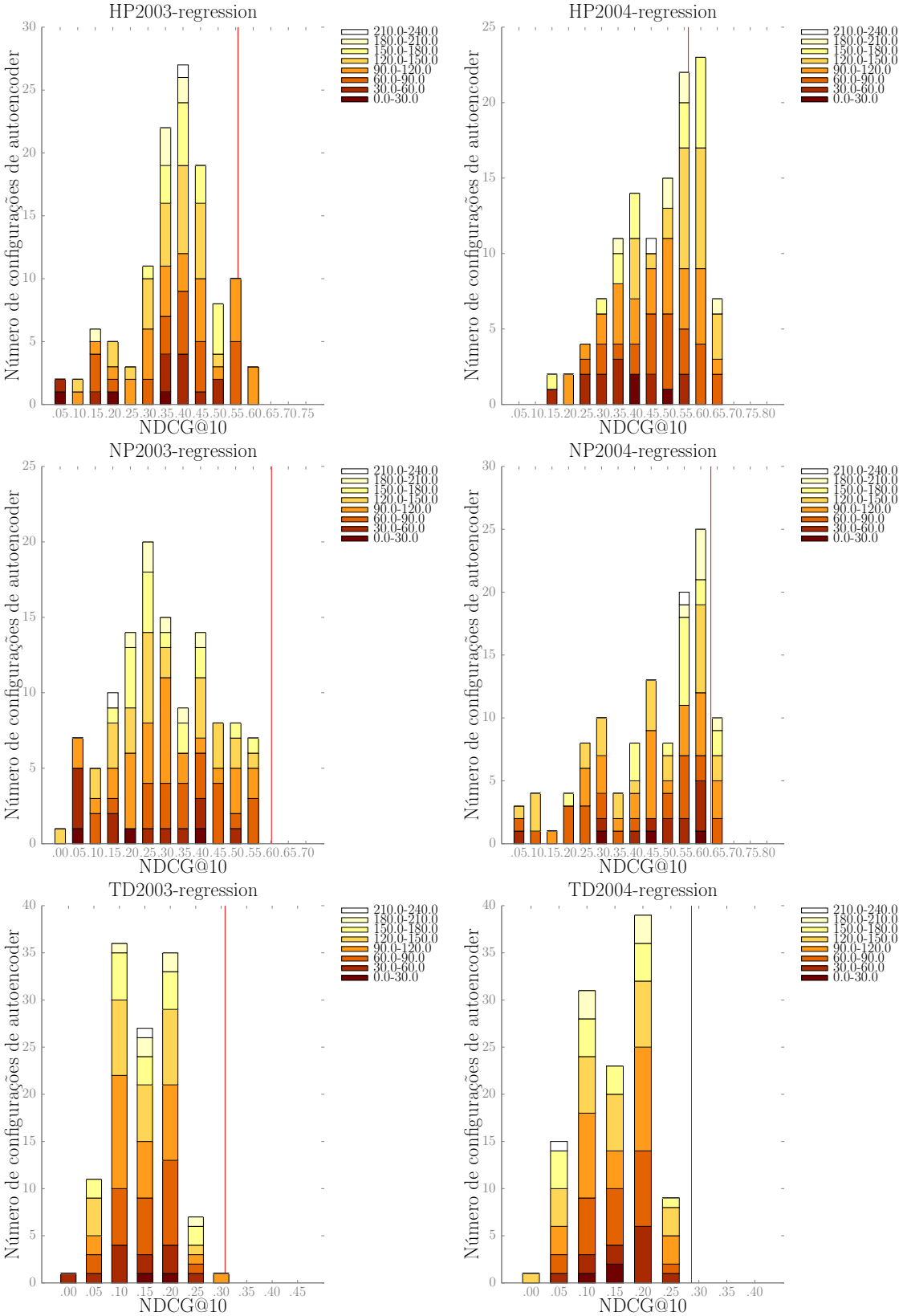

Figura 4.11. Regression - Desempenho na base GOV para a métrica NDCG@10. As cores das barras indicam a faixa de parametros daquele autoencoder, numa divisão por número total de nós.

4.5.6 Random Forests

Para o algoritmo Random Forests, vemos variações negativas fracas para as bases TD2004 e HP2004, e variações insignificantes pra as demais bases de dados. Vemos aqui novamente a tendência dos melhores resultados estarem na faixa de 120 a 210 nós totais, e é possível observar nas Figuras 4.12 e 4.13 que a variação nos resultados foi bem pequena: o desempenho dos autoencoders se concentrou em poucas faixas de resultados. Nenhuma taxa de corrupção prevaleceu entre os melhores resultados aqui.

Tabela 4.21. Desempenho dos autoencoders por numero total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo Random Forests, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.4%	2.29%	2.56%
30.0 a 60.0	9.46%	7.83%	11.77%
60.0 a 90.0	20.34%	17.59%	24.23%
90.0 a 120.0	25.42%	25.42%	25.43%
120.0 a 150.0	22.88%	24.34%	20.82%
150.0 a 180.0	13.56%	15.54%	10.75%
180.0 a 210.0	5.08%	6.02%	3.75%
210.0 a 240.0	0.85%	0.96%	0.68%
No. Combinações Estudadas	1416	830	586

Tabela 4.22. Desempenho dos autoencoders por nível de corrupção, contemplando somente a base OHSUMED. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo Random Forests, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

Nível de ruído nos dados	*Distribuição esperada*	*.10 ao topo*
0.0	50.03%	50.03%
0.3	49.97%	49.97%
No. Combinações Estudadas	2948	2948

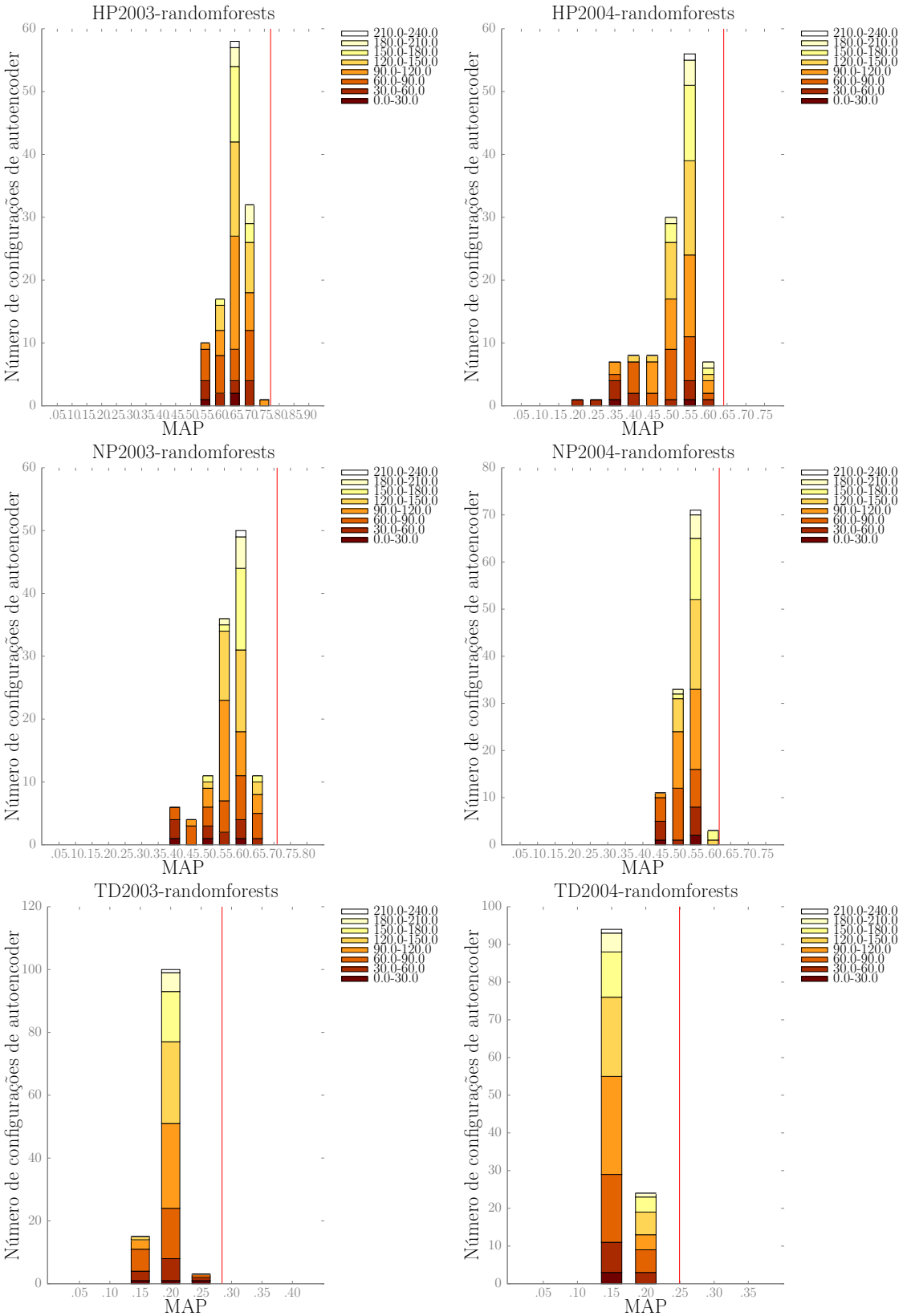

Figura 4.12. Random Forests - Desempenho para a métrica MAP na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós.

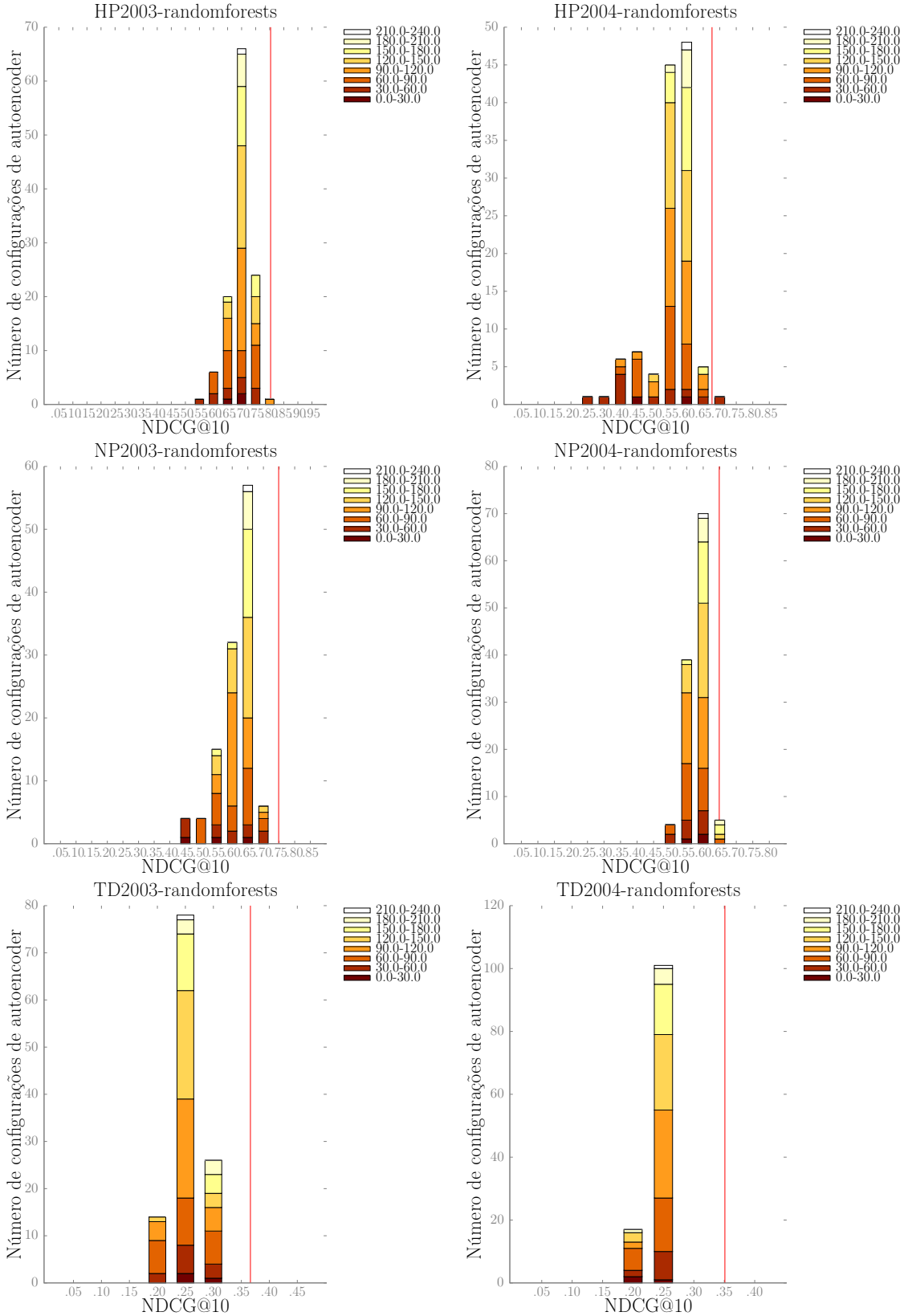

Figura 4.13. Random Forests - Desempenho para a métrica NDCG@10 na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós.

4.5.7 Mart

Houve variação positiva em relação à baseline, de 5 a 7 pontos percentuais, nas bases HP2004, NP2004 e TD2003. Nas bases HP2003 e NP2003 houve leve variação negativa, de 2.5 a 3 pontos percentuais. Além disso, não houve grande variação nos resultados, e a tendência de concentração dos melhores autoencoders se manteve na faixa de 120 a 180 nós. Nenhuma taxa de corrupção prevaleceu entre os melhores resultados aqui.

Tabela 4.23. Desempenho dos autoencoders por número total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo Mart, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.54%	3.02%	1.87%
30.0 a 60.0	9.6%	7.85%	12.07%
60.0 a 90.0	20.2%	18.0%	23.3%
90.0 a 120.0	25.28%	26.45%	23.64%
120.0 a 150.0	22.88%	24.03%	21.26%
150.0 a 180.0	13.56%	14.61%	12.07%
180.0 a 210.0	5.08%	5.31%	4.76%
210.0 a 240.0	0.85%	0.72%	1.02%
No. Combinações Estudadas	1416	828	588

Tabela 4.24. Desempenho dos autoencoders por nível de corrupção, contemplando somente a base OHSUMED. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo mart, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

Nível de ruído nos dados	*Distribuição esperada*	*.10 ao topo*
0.0	50.07%	50.07%
0.3	49.93%	49.93%
No. Combinações Estudadas	2948	2948

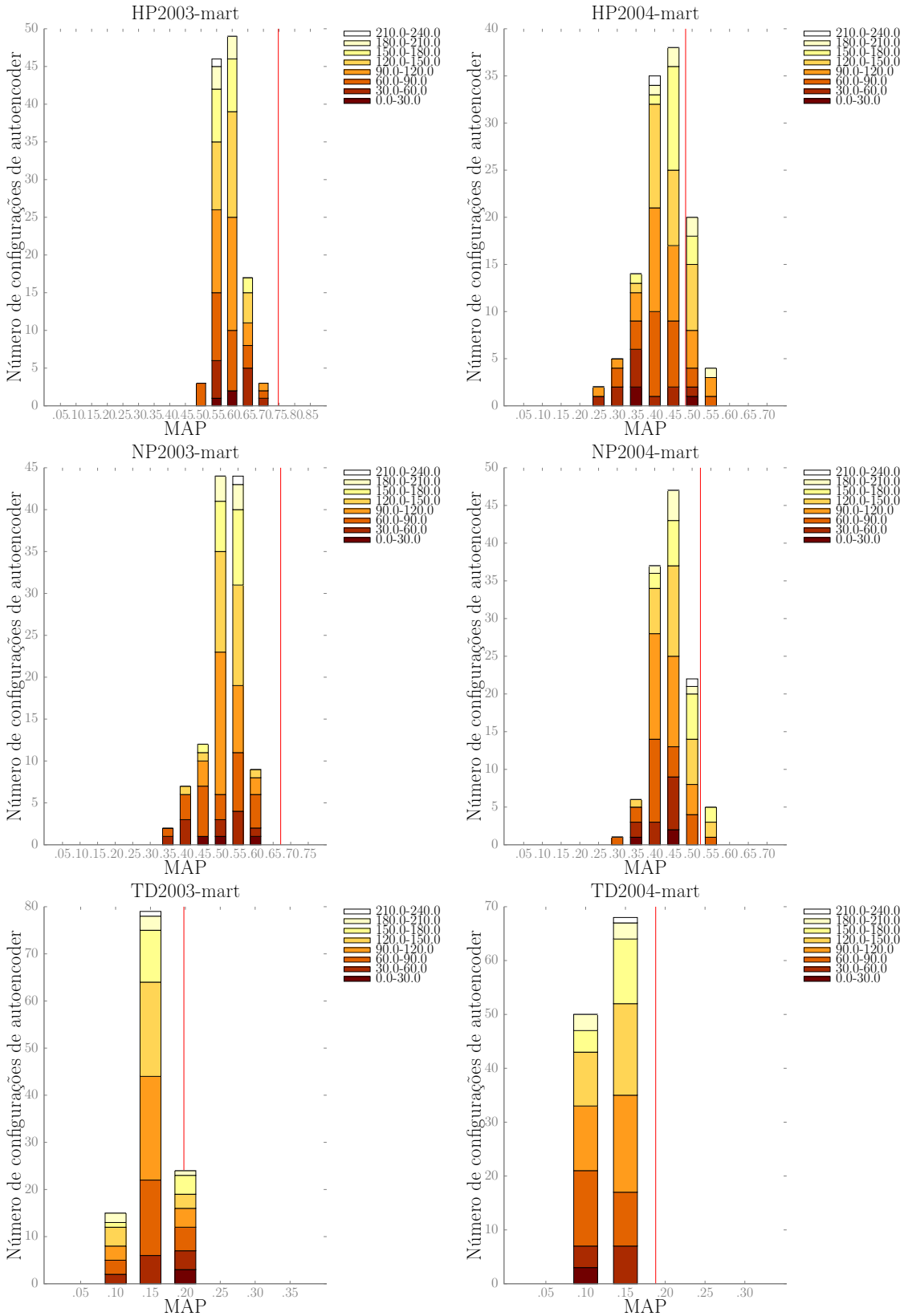

Figura 4.14. Mart - Desempenho para a métrica MAP na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós.

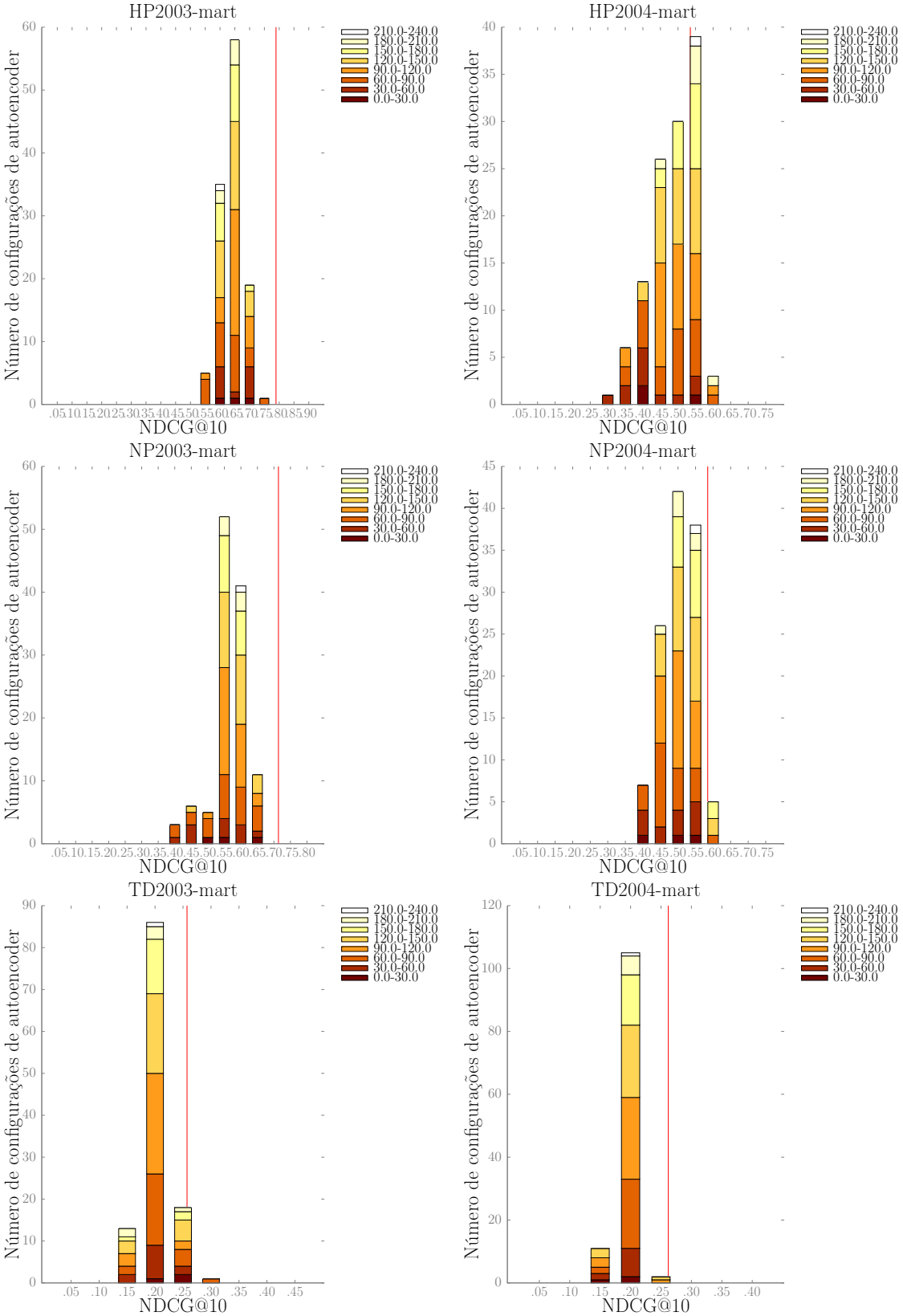

Figura 4.15. Mart - Desempenho para a métrica NDCG@10 na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós.

4.5.8 Lambdamart

Houve variação fortemente positiva, de 8 pontos percentuais, na base NP2004; fracamente positiva na base TD2003 e negativa na base TD2004, na faixa de 4 pontos percentuais. Não houve grande variação nos resultados dos autoencoders. Houve concentração maior que a esperada de bons resultados na faixa de 120 a 180 nós totais. Nenhuma taxa de corrupção prevaleceu entre os melhores resultados aqui.

Tabela 4.25. Desempenho dos autoencoders por número total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo Lambdamart, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.4%	2.55%	2.2%
30.0 a 60.0	9.53%	6.42%	13.87%
60.0 a 90.0	20.27%	16.85%	25.04%
90.0 a 120.0	25.42%	25.7%	25.04%
120.0 a 150.0	22.88%	25.33%	19.46%
150.0 a 180.0	13.56%	16.24%	9.81%
180.0 a 210.0	5.08%	5.82%	4.06%
210.0 a 240.0	0.85%	1.09%	0.51%
No. Combinações Estudadas	1416	825	591

Tabela 4.26. Desempenho dos autoencoders por nível de corrupção, contemplando somente a base OHSUMED. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para o algoritmo lambdamart, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

Nível de ruído nos dados	*Distribuição esperada*	*.10 ao topo*
0.0	50.03%	50.03%
0.3	49.97%	49.97%
No. Combinações Estudadas	2948	2948

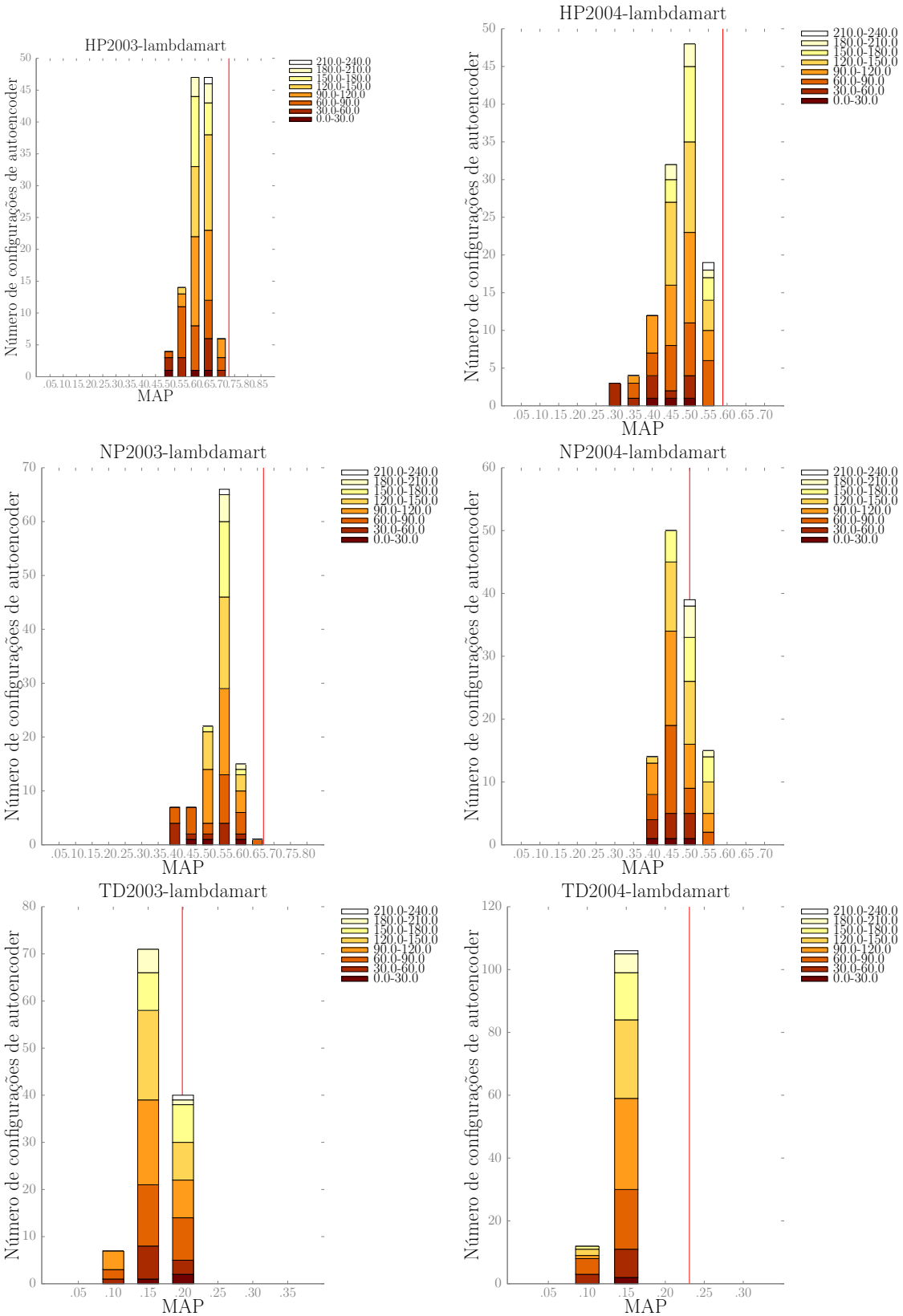

Figura 4.16. Lambdamart - Desempenho para a métrica MAP, na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós.

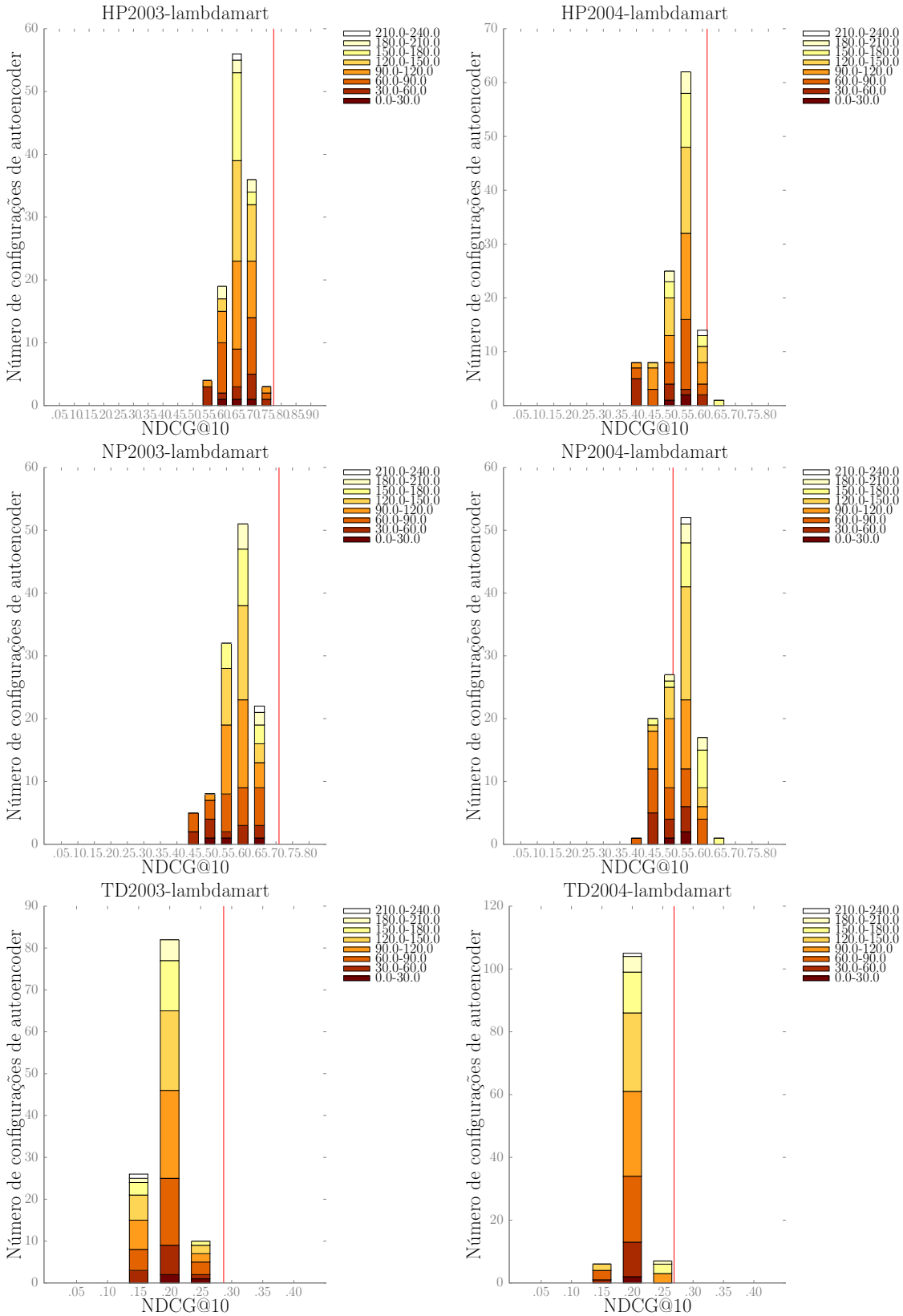

Figura 4.17. Lambdamart - Desempenho para a métrica NDCG@10, na base GOV. As cores das barras indicam a faixa de parâmetros daquele autoencoder, numa divisão por número total de nós.

4.6 Sumário

Para a base GOV, 120 autoencoders diferentes geraram representações cuja qualidade foi aferida por oito algoritmos de ordenação de respostas. A análise dos resultados trás uma série de conclusões interessantes:

Tabela 4.27. Desempenho dos autoencoders por numero total de nós. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para algum algoritmo, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

No. de Nós Totais (Grupos)	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0 a 30.0	2.4%	2.24%	2.75%
30.0 a 60.0	9.86%	8.58%	12.65%
60.0 a 90.0	20.18%	18.83%	23.12%
90.0 a 120.0	25.3%	25.26%	25.39%
120.0 a 150.0	22.79%	23.53%	21.18%
150.0 a 180.0	13.53%	14.79%	10.8%
180.0 a 210.0	5.08%	5.85%	3.42%
210.0 a 240.0	0.85%	0.93%	0.67%
No. Combinações Estudadas	11328	7764	3564

Tabela 4.28. Desempenho dos autoencoders por nível de corrupção, contemplando somente a base OHSUMED. A **coluna do meio** representa o conjunto dos autoencoders que ficaram a uma distância de 10 pontos percentuais, no máximo, do melhor resultado obtido para algum algoritmo, e a **coluna da direita** os autoencoders que ficaram abaixo disso. A **coluna da esquerda** mostra a distribuição dos autoencoders de acordo com seu número total de nós.

Nível de ruído nos dados	*Distribuição esperada*	*.10 ao topo*	*.00 a .15*
0.0	50.02%	49.7%	51.7%
0.3	49.98%	50.3%	48.3%
No. Combinações Estudadas	25074	21142	3932

1. **O parâmetro "Número de Nós Total" tem faixas que tendem a apresentar resultados melhores, mas qualquer faixa pode dar resultados**

ruins: Para cada resultado de ordenação de respostas há um espectro de resultados que vai de números muito ruins a números muito bons. Todas as faixas de "números de nós totais" estudadas obtiveram resultados que, consistentemente, estavam distribuídas ao longo de todo o espectro de resultados possíveis. Ou seja: nenhuma quantidade de nós totais é imune a resultados ruins. Qualquer valor que se escolha para esse parâmetro do autoencoder está sujeito a resultados insatisfatórios. Há, entretanto, uma clara tendência de se encontrar resultados melhores numa faixa específica de valores para esses parâmetros. Isso significa que escolher uma faixa específica de valores para o número de nós total não dá garantia de bons resultados, mas certamente aumenta suas chances de encontrar bons resultados.

2. **Há leve tendência de se encontrar os melhores resultados na faixa de autoencoders com 120 a 180 nós totais:** Na grande maioria das análises, vemos que a faixa de 120 a 180 nós totais tendeu a apresentar bons resultados. Tal tendência, embora não absoluta, não foi influenciada de forma relevante pela métrica estudada, e foi levemente influenciada pelo algoritmo estudado. Na base GOV, fosse a base HP, NP ou TD, os resultados se mantiveram consistentes, e variou de forma relevante no caso da base OHSUMED. Dentre todos os algoritmos estudados essa tendência de resultados se apresentou, exceção feita aos algoritmos Regression e Rankboost, onde não houve tendência de apresentação de melhores resultados em nenhuma faixa. Nos algoritmos Adarank, Listnet e Random Forests, a faixa de 180 a 210 nós seguiu tendência similar. Quase todas as bases de dados estudadas também apresentaram essa tendência, exceção feita à base de dados TD, uma base notóriamente difícil de se classificar, onde a tendência se restringiu à faixa de 150 a 210 nós. Vemos, em suma, que independente de métrica, base de dados ou algoritmo, é possível afirmar com razoável segurança que a faixa de nós entre 120 e 180 tende a apresentar os melhores resultados.

3. **É muito difícil, mas possível, melhorar o resultado dos algoritmos para algumas bases de dados:** Não houve algoritmo que obteve melhora ou piora em relação à baseline para todas as bases de dados estudadas. E, em todos os casos onde houve, a quantidade de representações que superaram a da baseline foi muito pequena, ainda que a quantidade de representações quase boas possa ter sido por vezes, bem grande. Isso mostra que melhorar, para determinado algoritmo e determinada base, uma representação utilizando um autoencoder é bem difícil - o que ressalta a importância de se ter bons parâmetros estáticos de autoencoder para limitar o seu espaço de busca por configurações boas.

4. **As novas representações das bases de dados influenciam cada algoritmo de maneira diferente:** Observamos que, enquanto os algoritmos Adarank, Rankboost, Listnet, Ranknet e Regression tiveram desempenhos muito variáveis sobre as representações, os algoritmos Random Forests, Mart e Lambdamart variaram muito menos. As representações oferecem os mesmos dados codificados para todos os algoritmos, mas a maneira como cada um lida com eles varia muito. Para vários, algumas representações simplesmente perdem muita informação, enquanto para outros é possível tratar todos os casos de forma muito mais semelhante. Cada algoritmo, em suma, reage aos dados apresentados de maneira diferente.

5. **Não há garantia de que uma representação específica de base de dados terá resultados semelhantes para todos os algoritmos:** Isso decorre da conclusão do item acima. Uma mesma representação gerada por um autoencoder pode oferecer uma melhora razoável para um algoritmo e uma piora significativa para outro. A eficácia das representações está intrínsicamente ligada ao algoritmo que a utiliza.

6. **A métrica utilizada, seja MAP ou NDCG@10, não influencia no desempenho das representações na ordenação:** Vimos que, para o parâmetro estático "Número de Nós Total", a tendência de onde se encontrar os melhores resultados foi práticamente a mesma. Isso indica que a escolha da métrica não afeta o desempenho do algoritmo ou da representação que ele utiliza.

7. **Inserir ruído nos autoencoders não melhorou os resultados:** Salvo exceções, onde a tendência de melhora foi sutil e variou, na maior parte dos casos nenhuma taxa de corrupção apresentou tendência de ter resultados melhores.

   Ou seja: as representações alternativas criadas neste trabalho para aprimorar tarefas de Learning to Rank se mostraram ao mesmo tempo promissoras e de difícil trato. Embora seja difícil, cada algoritmo apresentou melhora relevante em algum dos seus resultados, quando executado sobre as representações criadas. O parâmetro "Número de Nós Total" apresentou tendência de dar melhores resultados quando na faixa de 120 a 150 nós totais - isso para todas as bases de dados, sejam elas as bases GOV, de 44 atributos, ou a base OHSUMED, de 64 atributos. A base GOV repetiu tal tendência para a faixa de 150 a 180, e a base OHSUMED a repetiu para a faixa de 90 a 120 nós totais. A métrica de avaliação não influencia no desempenho das representações, e tanto as bases de dados quando os algoritmos, por si só, influenciam pouco tal tendência.

Além disso, vemos que o fator mais crucial para se determinar o potencial de melhora de uma representação criada por autoencoder é o par algoritmo-base de dados. Cada par interage de maneira única com as representações: uma mesma representação pode ser boa para um e ruim para outro. Isso mostra que, mais do que simplesmente melhorar as bases de dados de forma geral, as representações agem adaptando uma base de dados a um algoritmo, e isso gera resultados positivos.

Vemos, portanto, que o aprimoramento das bases de dados, o estudo de seus atributos e de suas representações resulta numa melhora de desempenho que, por vezes, reduz a diferença de desempenho de dois algoritmos muito diferentes. Disso, conclue-se que quanto melhor uma base de dados for, menos relevante o algoritmo de ordenação de respostas tende a ser. O estudo das bases de dados, e não mais dos algoritmos de ordenação de respostas, mostra-se interessante como forma de buscar superar o atual estado da arte em ordenamento de documentos.

Capítulo 5

Conclusão e Trabalhos Futuros

Neste trabalho, nós estudamos o aprendizado de representações para tarefas de ordenação de respostas. Nós utilizamos Stacked Autoencoders para gerar diferentes representações de uma base de dados referência e analisamos a forma como tais representações influenciavam os resultados de algoritmos de ordenamento de documentos tradicionais. Estudamos ainda se um parâmetro estático dos autoencoders, o número de nós total, teria uma faixa paramétrica limitada onde melhores resultados de representações tenderiam a se apresentar.

Nós observamos que o aprendizado de representações das bases de dados é uma forma difícil, mas muito eficaz, de se melhorar os resultados apresentados pelos algoritmos de ordenação de respostas. Vemos que boas representações de bases de dados reduzem a diferença de desempenho entre algoritmos vastamente diferentes, o que indica que, se uma base de dados for boa o surficiente, o algoritmo de ordenamento utilizado importa pouco na obtenção de um ótimo desempenho.

Mostramos ainda que, embora os autoencoders sejam uma forma difícil de se conseguir representações melhores das bases de dados, há uma faixa de nós totais que, nas bases observadas, tendeu a apresentar os melhores resultados. Observamos que diferentes pares algoritmo-base de dados obtém desempenhos variáveis sobre uma mesma representação, o que indica que tais representações funcionam mais como uma forma de adaptar uma base de dados a um algoritmo do que como forma geral de melhorar as bases como um todo. Ou seja: a inclusão de uma etapa de tratamento dos dados em algorimos de ordenamento é uma forma de se melhorar resultados que se apresenta muito promissora.

Diversas perguntas surgiram dos resultados analisados, que se apresentam como boas linhas de trabalhos futuros. Vemos que cada par algoritmo-base de dados reage às mesmas representações de formas diferentes. Surgem então as perguntas: que fatores

determinam essa reação? É possível delimitar um conjunto de parâmetros bons para o autoencoder para diferentes pares? Aqui, estudamos também que o parâmetro "Número de nós total" tem uma faixa que apresenta melhores resultados. Outros parâmetros estáticos do autoencoder podem ser estudados de forma semelhante, como corrupção ou largura. Quanto mais delimitada a faixa que pode trazer melhores resultados for, mais interessante o autoencoder se apresenta, enquanto forma de melhorar os resultados dos algoritmos de ordenação de respostas. Vemos ainda tais parâmetros do autoencoder não possuem tendências absolutas: diferentes bases de dados e algoritmos apresentam tendências por vezes divergentes às gerais. Porque isso ocorre?

Em suma, este trabalho mostra que aprimorar as bases de dados é uma forma muito promissora de se melhorar os resultados dos algoritmos de ordenamento, em oposição a simplesmente aprimorar o algoritmo de ordenação de respostas em si. Vemos que o autoencoder é uma forma boa de trazer tais melhoras, havendo faixas de parâmetros que claramente trazem melhores resultados. Diversas questões aqui levantadas se apresentam como boas linhas para trabalhos futuros.

Referências Bibliográficas

[1] Amiri, H.; Resnik, P.; Boyd-Graber, J. & III, H. D. (2016). Learning text pair similarity with context-sensitive autoencoders. Em *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pp. 1882--1892.

[2] Bengio, Y. & Lee, H. (2015). Editorial introduction to the neural networks special issue on deep learning of representations. *Neural Networks*, 64:1--3.

[3] Bergstra, J.; Breuleux, O.; Bastien, F.; Lamblin, P.; Pascanu, R.; Desjardins, G.; Turian, J.; Warde-Farley, D. & Bengio, Y. (2010). Theano: A cpu and gpu math compiler in python. Em *Proc. 9th Python in Science Conf*, pp. 1--7.

[4] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5--32.

[5] Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N. & Hullender, G. (2005a). Learning to rank using gradient descent. Em *Proceedings of the 22nd international conference on Machine learning*, pp. 89--96. ACM.

[6] Burges, C. J. C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N. & Hullender, G. N. (2005b). Learning to rank using gradient descent. Em *Proc. of the 22nd Intl. Conf. on Machine Learning*, pp. 89--96.

[7] Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F. & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. Em *Proceedings of the 24th international conference on Machine learning*, pp. 129--136. ACM.

[8] Chapelle, O.; Chang, Y. & Liu, T.-Y. (2011). Future directions in learning to rank. Em *Yahoo! Learning to Rank Challenge*, pp. 91--100.

[9] Chen, M.; Weinberger, K. Q.; Sha, F. & Bengio, Y. (2014). Marginalized denoising auto-encoders for nonlinear representations. Em *ICML*, pp. 1476--1484.

[10] Crammer, K.; Singer, Y. et al. (2001). Pranking with ranking. Em *Nips*, volume 1, pp. 641--647.

[11] Dang, V. (2013). Ranklib.

[12] Dela Rosa, K.; Metsis, V. & Athitsos, V. (2012). Boosted ranking models: a unifying framework for ranking predictions. *Knowledge and information systems*, 30(3):543--568.

[13] Ding, W.; Geng, X. & Zhang, X. (2015). Learning to rank from noisy data. *ACM TIST*, 7(1):1.

[14] Duh, K. & Kirchhoff, K. (2008). Learning to rank with partially-labeled data. Em *Proc. of the 31st ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 251--258.

[15] Duh, K. & Kirchhoff, K. (2011). Semi-supervised ranking for document retrieval. *Computer Speech & Language*, 25(2):261--281.

[16] Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.-A.; Vincent, P. & Bengio, S. (2010a). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625--660.

[17] Erhan, D.; Courville, A. C.; Bengio, Y. & Vincent, P. (2010b). Why does unsupervised pre-training help deep learning? Em *Proc. of the 13th Intl. Conf. on Artificial Intelligence and Statistics*, pp. 201--208.

[18] Freund, Y.; Iyer, R.; Schapire, R. E. & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933--969.

[19] Geng, X.; Liu, T.-Y.; Qin, T. & Li, H. (2007). Feature selection for ranking. Em *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 407--414. ACM.

[20] Gigli, A.; Lucchese, C.; Nardini, F. M. & Perego, R. (2016). Fast feature selection for learning to rank. Em *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval*, pp. 167--170. ACM.

[21] Hinton, G. E.; Krizhevsky, A. & Wang, S. D. (2011). Transforming auto-encoders. Em *International Conference on Artificial Neural Networks*, pp. 44--51. Springer.

[22] Hua, G.; Zhang, M.; Liu, Y.; Ma, S. & Ru, L. (2010). Hierarchical feature selection for ranking. Em *Proceedings of the 19th international conference on world wide web*, pp. 1113--1114. ACM.

[23] Järvelin, K. & Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422--446.

[24] Krizhevsky, A. & Hinton, G. E. (2011). Using very deep autoencoders for content-based image retrieval. Em *ESANN*.

[25] Lai, H.-J.; Pan, Y.; Tang, Y. & Yu, R. (2013). Fsmrank: Feature selection algorithm for learning to rank. *IEEE transactions on neural networks and learning systems*, 24(6):940--952.

[26] Laporte, L.; Flamary, R.; Canu, S.; Déjean, S. & Mothe, J. (2014). Nonconvex regularizations for feature selection in ranking with sparse svm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(6):1118--1130.

[27] Li, P.; Wu, Q. & Burges, C. J. (2007). Mcrank: Learning to rank using multiple classification and gradient boosting. Em *Advances in neural information processing systems*, pp. 897--904.

[28] Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225--331.

[29] Mesnil, G.; Bordes, A.; Weston, J.; Chechik, G. & Bengio, Y. (2014). Learning semantic representations of objects and their parts. *Machine learning*, 94(2):281--301.

[30] Olshausen, B. A. et al. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607--609.

[31] Page, L.; Brin, S.; Motwani, R. & Winograd, T. (1999). The pagerank citation ranking: bringing order to the web.

[32] Pahikkala, T.; Tsivtsivadze, E.; Airola, A.; Boberg, J. & Salakoski, T. (2007). Learning to rank with pairwise regularized least-squares. Em *SIGIR 2007 workshop on learning to rank for information retrieval*, volume 80, pp. 27--33. Citeseer.

[33] Pan, F.; Converse, T.; Ahn, D.; Salvetti, F. & Donato, G. (2009). Feature selection for ranking using boosted trees. Em *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 2025--2028. ACM.

[34] Pan, Y.; Luo, H.; Qi, H. & Tang, Y. (2011). Transductive learning to rank using association rules. *Expert Syst. Appl.*, 38(10):12839--12844.

[35] Qin, T.; Liu, T.-Y.; Xu, J. & Li, H. (2010). Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346--374.

[36] Rigutini, L.; Papini, T.; Maggini, M. & Scarselli, F. (2011). Sortnet: Learning to rank by a neural preference function. *IEEE Trans. Neural Networks*, 22(9):1368--1380.

[37] Robertson, S. & Zaragoza, H. (2009). *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

[38] Salton, G. & Yang, C.-S. (1973). On the specification of term values in automatic indexing. *Journal of documentation*, 29(4):351--372.

[39] Schölkopf, B.; Smola, A. J. & Müller, K. (1997). Kernel principal component analysis. Em *Proc. of the 7th Intl. Conf. on Artificial Neural Networks*, pp. 583--588.

[40] Sculley, D. (2010). Combined regression and ranking. Em *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 979--988. ACM.

[41] Severyn, A. & Moschitti, A. (2015). Learning to rank short text pairs with convolutional deep neural networks. Em *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 373--382. ACM.

[42] Simon, P. (2013). *Too Big to Ignore: The Business Case for Big Data*, volume 72. John Wiley & Sons.

[43] Tax, N.; Bockting, S. & Hiemstra, D. (2015). A cross-benchmark comparison of 87 learning to rank methods. *Information processing & management*, 51(6):757--772.

[44] Taylor, M.; Guiver, J.; Robertson, S. & Minka, T. (2008). Softrank: optimizing non-smooth rank metrics. Em *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pp. 77--86. ACM.

[45] Veloso, A. A.; Almeida, H. M.; Gonçalves, M. A. & Meira Jr, W. (2008). Learning to rank at query-time using association rules. Em *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 267--274. ACM.

[46] Vincent, P.; Larochelle, H.; Bengio, Y. & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. Em *Proceedings of the 25th international conference on Machine learning*, pp. 1096--1103. ACM.

[47] Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y. & Manzagol, P. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371--3408.

[48] Wang, L.; Lin, J. J. & Metzler, D. (2010). Learning to efficiently rank. Em *Proceeding of the 33rd ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 138--145.

[49] Wu, Q.; Burges, C. J.; Svore, K. M. & Gao, J. (2010). Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254--270.

[50] Xia, F.; Liu, T.; Wang, J.; Zhang, W. & Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. Em *Proc. of the 25th Intl. Conf. on Machine Learning*, pp. 1192--1199.

[51] Xu, J. & Li, H. (2007). Adarank: a boosting algorithm for information retrieval. Em *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 391--398. ACM.