Nome: Rafael Claro Ito (R.A.: 118430)
Resumo do artigo: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

Neste artigo os autores exploram diferentes técnicas de transfer learning em NLP e propõem um *framework* unificado que abrange todos os problemas de linguagem na forma de *text-to-text*, isto é, tanto a entrada como a saída do modelo são textos. Este modelo proposto, denominado **T5** (***T**ext-**t**o-**T**ext **T**ransfer **T**ransformer*), juntamente com um novo *dataset* denominado **C4**, os levaram a atingir resultados estado-da-arte em diferentes *benchmarks*, como sumarização, perguntas e respostas, classificação de textos, entre outros.

C4 (Colossal Clean Crawled Corpus):
Este novo *dataset* é baseado conjunto de dados *Common Crawl*, que é uma coleção de textos extraídos da *web* onde são removidas *markups* e outros conteúdos que não são textos. No entanto, como este *dataset* original é bastante "poluído", os autores fizeram uma série de filtragem em cima dele para obter uma versão mais enxuta e limpa. Dentre as modificações, podemos destacar: remoção de páginas com qualquer palavra de conteúdo obsceno, remoção de *warnings* JavaScript e códigos, manter apenas linhas que terminam com sinal de pontuação, seleção de idioma, etc.

Na seção 3 do artigo (***Experiments***) os autores tentam comparar diversas abordagens diferentes. Para isso, nem sempre é possível usar um modelo replicado e engessado para as tarefas. Por exemplo, pelo fato de o BERT ser um modelo que apresenta apenas um *encoder*, não possuindo portanto um *decoder*, isso o faz ser aplicável em tarefas como classificação ou predição de trechos de texto, mas não para tarefas generativas, como tradução ou sumarização. Assim, as comparações que são feitas se baseiam em modelos análogos, por exemplo usando a tarefa MLM para representar o modelo do BERT.

***Unsupervised objective*:**
Nesta subseção os autores descrevem como foi feito o pré-treinamento do modelo a partir de dados não rotulados com o objetivo de ensinar ao modelo conhecimentos generalizáveis que virão a ser úteis nas tarefas seguintes. Aqui, são extraídos 15% dos *tokens* dos textos de entrada, sendo estes substituídos por um *token* sentinela (caso um ou mais *tokens* adjacentes também estejam mascarados, todos recebem o mesmo *token* sentinela). Na fase de treinamento ocorre uma inversão para a saída, isto é, o texto original não mascarado recebe o símbolo do *token* sentinela, e os *tokens* mascarados na entrada apresentam seu texto íntegro na saída. Treina-se o modelo nesta tarefa de *denoising* para predizer essa saída seguida de um *token* sentinela final.

***Attention masks*:**
São apresentados três tipos de máscara. A primeira é uma máscara totalmente visível, que permite que o mecanismo de atenção olhe para todos os elementos de entrada na hora de gerar uma saída. A segunda é uma máscara causal, onde para a saída *i*, só é permitido levar em conta as entradas anteriores à *i*. Por fim, a última máscara que é uma mistura das duas primeiras é denominada máscara causal com prefixo, isto é, para uma parcela da entrada o mecanismo de atenção é totalmente visível, para o restante é uma máscara causal.

***Arquiteturas*:**
Aqui são mostradas as arquiteturas das variações do *Transformer* que serão analisadas e comparadas (figura 4). A primeira arquitetura é um ***encoder-decoder***, que usa uma máscara totalmente visível no *encoder* e causal no *decoder*. A segunda é um **modelo de linguagem** que usa somente a máscara causal, onde o modelo é treinado para predizer o próximo elemento. No contexto *text-to-text*, podemos definir como entrada para o modelo a tarefa a ser realizada, por exemplo tradução, mais a entrada efetiva, e saída seria o texto traduzido. Por fim, a terceira arquitetura é um ***prefix LM***, que usa o terceiro tipo de máscara e tenta cobrir desvantagens do tradicional modelo de linguagem. É feita uma comparação interessante mostrando como este tipo de arquitetura lembra o BERT, porém integrando a camada de saída no *decoder* do *Transformer*.

Comparando estruturas de modelos diferentes:
Aqui são comparados 5 tipos de modelos para a tarefa de *denoising* e para a tarefa autorregressiva (*language model)*. São comparados os número de parâmetros (P ou 2P) e o número de FLOPS (M ou M/2) representando o custo computacional em diversas tarefas (GLUE, SQuAD, tradução de inglês para alemão/francês/romeno, etc). Os modelos comparados foram: *encoder-decoder* tradicional, com parâmetros compartilhados entre *encoder* e *decoder* e com metade do número de camadas, além do modelo de linguagem e de um *prefix LM*.

Resultados:
Os resultados são mostrados na tabela 2, sendo o *encoder-decoder* o modelo que obteve o melhor desempenho. Vale notar que mesmo o modelo reduzido, com metade do número de parâmetros, performa melhor do que o modelo de linguagem. Também é confirmada a concepção geral de que usando o objetivo de *denoising* resulta em melhores desempenhos em tarefas *downstream* do que objetivos de modelo de linguagem. Por fim, os autores constatam que uma estrutura *encoder-decoder* pode atingir resultados satisfatórios em tarefas generativas e tarefas de classificação.