Identificação e classificação de comentários tóxicos utilizando processamento de linguagens naturais e técnicas de aprendizagem profunda

Miguel Angelo Cece de Castro Neto[1], **Paulo Alves dos Santos Junior**[1]

[1] Departamento de Informática – Universidade Estadual de Ponta Grossa (UEPG)
Avenida General Carlos Cavalcanti, 4748
CEP 84030-900 – Ponta Grossa, PR – Brasil

Abstract. *The article addresses the problem of multiclass sentiment analysis at the sentence level. Recurrent neural networks and their demonstrations have demonstrated successful modeling of sentiment classifiers and last generation in language modeling with the same demonstration demonstrating efficiency and performance in various tasks. The datasheet of the learned paper transfer, larger database, has proven effective in small databases. This work proposes the use ages between architecture in a with modeling of language to a classification of the sentiment in comment.*

Resumo. *Este artigo aborda o problema da análise de sentimentos multiclasse no nível de sentença. Redes neurais recorrentes e suas variações demonstraram sucesso na modelagem de classificadores de sentimentos e recentemente a modelagem de linguagens através dessa arquitetura demonstrou-se eficaz atingindo o estado da arte em várias tarefas. O uso da técnica de transferência de aprendizado, proveniente de banco de dados maiores, se demonstrou eficiente em banco de dados pequenos. Nesse trabalho propomos o uso destas variações arquitetura em conjunto com a modelagem de linguagem para a classificação de sentimento em comentários.*

1. Introdução

Discutir assuntos na internet pode ser difícil. Ameaças, ofensas pessoais, e assédio, podem criar um ambiente tóxico dentro de fóruns e redes sociais, e até afastar usuários. As plataformas lutam para combater efetivamente discussões tóxicas e limitar ou desligar usuários com essas práticas.

Recentemente, em resposta ao crescimento do uso de redes sociais, comentários são publicados massivamente. Como a escrita compõe grande parte de todos os dados gerados pela humanidade, que até então, apenas serviam para consultas, hoje servem como base de dados para o desenvolvimento de sistemas de processamento de linguagens naturais. Aplicando técnicas de aprendizagem profunda e processamento de linguagens naturais podemos automatizar processos análise e classificação de textos com uma boa taxa de acerto, comparado à classificação por métodos bayesianos, por exemplo.

No problema de classificação de comentários tóxicos, basicamente a análise de sentimentos é utilizada para representar em quais classes de toxicidade a pertence. Formalmente, dada a sentença s, o objetivo da análise de sentimentos é extrair a seguinte sêxtupla:

A análise de sentimentos é determinada comumente de maneira binária (positiva e negativa), porém abordagens multiclasse também são possíveis. Em uma abordagem multiclasse determinado comentário pode possuir diversos rótulos simultaneamente. Nesse artigo exploraremos tal abordagem e utilizaremos redes neurais recorrentes em conjunto com estruturas celulares específicas, como células recorrentes bloqueadas (Cho et al. 2014) e células de longa memória de curto termo (Hochreiter Schmidhuber. 1997). Abordaremos técnicas de transferência de aprendizado e construção de modelos de linguagem.

A utilização de redes neurais artificiais voltou a popularizar-se no reconhecimento de padrões em imagens devido a evolução de hardware, tais métodos de aprendizagem de máquina combinados ao processamento de linguagens naturais, permitem a avaliação automática de padrões e extração de conhecimento de bases de texto. A identificação de comentários ofensivos em textos é um derivado do caso geral de análise de sentimentos, na era da internet, identificar e classificar sentenças permite tomada de decisões e maior controle e filtragem de conteúdo.

2. Trabalho Relacionado

A análise de sentimentos está sob grande aprimoramento devido a aplicação de redes recorrentes (Karpathy. 2015), redes convolucionais (Lecun et al. 1998) e transferência de aprendizado (Howard Ruder. 2018). Antes dessa abordagem, tal tarefa era realizada através de métodos utilizando contagem, ou extração de características processadas de maneira esparsa, para classificação utilizando algum método superficial de aprendizado, como por exemplo, máquinas de vetores de suporte (Vapnik et al. 1995), naive bayes e naive bayes svm (Wang and Manning. 2012).

O principal problema do método superficial citado acima, é que ele não é capaz de identificar o contexto de palavras, não sendo uma opção robusta para classificação.

Redes recorrentes e suas variações são aplicadas utilizando comumente matrizes que definem significado próximos ao real para cada palavra, como por exemplo Word2Vec (Mikolov et al. 2013) ou GloVe (Socher et al. 2014). Tais métodos provem contexto utilizando palavras próximas (modelo skip-gram) para modelas matrizes e consequentemente contexto para redes recorrentes.

Transferência de aprendizado de bancos de dados maiores é um desafio na área de processamento de linguagens naturais. Trabalhos recentes propõem a utilização de modelos de linguagem universais, que sofrerão afinação (fine tuning) para atender uma tarefa específica (Howard Ruder. 2018) com a possibilidade de utilização de dados não supervisionados para gerar um aumento de performance (Radford et al. 2018). A utilização de dados não rotulados permite reduzir drasticamente a necessidade de exemplos rotulados. O estado da arte da maioria das tarefas na área de linguagens naturais está relacionado ao uso de transferência de aprendizado e aprendizado não supervisionado, que é um dos maiores desafios atualmente.

2.1. Funções de ativação

Funções de ativação não lineares, dão capacidades não lineares para redes neurais (Lecun et al. 1998).

2.1.1. ReLU - Rectified Linear Activation Function

Dada pela fórmula: FORMULA

A imagem abaixo descreve expressão da ReLU em um intervalo de 5 a –5.

IMAGEM RELU

É a função de otimização padrão recomendada para uso na maioria das redes neurais. Aplicando essa função no output de uma transformação linear produz uma transformação não linear. A função permanece muito próxima de ser linear, contudo, no sentido de que é uma função linear por partes com duas peças lineares. Como as unidades lineares retificadas são quase lineares, elas preservam as propriedades que tornam os modelos lineares fáceis de otimizar com métodos baseados em gradiente. Eles também preservam muitas das propriedades que fazem modelos lineares generalizarem sistemas a partir de componentes mínimos (Goodfellow et al. 2017).

2.1.2. Sigmoide

A função sigmoide é dada pela formula: FORMULA

A imagem abaixo descreve a curva sigmoidal em um intervalo de 5 a –5.

IMAGEM SIGMOIDE

Aplicando essa função no output de uma transformação linear, teremos como resultado um valor entre 0 e 1.

Além de a função sigmoide ser utilizada na estrutura de células de longa memória de curto termo e células recorrentes bloqueadas, também pode ser usada na saída do modelo. Poderíamos por exemplo, na classificação de sentimentos em texto, considerar valores próximos de 0 um sentimento negativo, e próximos de 1, positivo. Os valores maiores iguais a 0.5 serão classificados como positivos, enquanto 0.5 serão classificados como negativos.

2.1.3. Tangente Hiperbólica

Dada pela formula: FORMULA

A imagem abaixo descreve a curva da tangente hiperbolica em um intervalo de 5 a -5.

Produz um resultado entre 1 e -1, e é usada na estrutura de células simples de redes recorrentes, células de longa memória de curto termo e células recorrentes bloqueada. Em comparação a função de ativação sigmoide, possui a vantagem de ser simétrica em relação ao eixo x.

2.2. Otimizadores

2.2.1. Escolhendo o Algoritmo de Otimização

Infelizmente ainda não existe um consenso sobre qual é o melhor algoritmo de otimização. Schaul et al. (2014) apresentou uma comparação valiosa com um grande número de algoritmos de otimização através de uma ampla variedade de tarefas de otimização. Enquanto os resultados sugeriam que a família de algoritmos com taxa de aprendizado adaptativa (como RMSProp e AdaDelta) apresentavam desempenho robusto, não houve um único algoritmo que emergiu sobre os outros.

Atualmente, os algoritmos de otimização mais populares são Gradiente Descendente Estocástico, Gradiente Descendente Estocástico com Momento, RMSProp, RMSProp com Momento, AdaDelta e Adam. A escolha de qual algoritmo de otimização usar, parece depender mais na familiaridade do desenvolvedor com o com o algoritmo (para afinar os hiperparametros mais facilmente) (Goodfellow et al. 2017).

2.2.2. Otimizador Adam

Com a utilização de minibatches, o progresso no gradiente tende a oscilar, logo o aprendizado pela rede se torna lento. Várias soluções foram propostas para solucionar esse problema, e a mais popular é uma junção das anteriores. O otimizador Adam (Kingma Jimmy, 2014) utiliza a estimação do primeiro e do segundo momento dos gradientes.

2.3. Função de Custo

Um aspecto importante no desenvolvimento de redes neurais profundas é na escolha de função de custo. Felizmente, as funções de custo para redes neurais são mais ou menos as mesmas da de outros modelos paramétricos, assim como modelos lineares(Goodfellow et al. 2017).

2.3.1. Entropia Cruzada

O cálculo do custo por entropia cruzada é dado pela seguinte formula: FORMULA

2.4. Regularização

Um problema central em aprendizado de máquina é como fazer um algoritmo que irá ter um bom desempenho não somente nos dados de treino, mas também em novos dados inseridos nele. Muitas estratégias usadas em aprendizado de máquina são explicitamente desenvolvidas para reduzir o erro no conjunto de teste, possivelmente à custa de um erro maior no conjunto de treino. Essas estratégias são conhecidas coletivamente como regularização (Goodfellow et al. 2017).

O Dropout oferece pouco custo computacional, porém um método poderoso para regularização de uma ampla família de modelos (Srivastava et al. 2014).

Especificamente, dropout treina o conjunto consistindo através de sub-redes geradas pela remoção de unidades que fazem parte das camadas intermediarias ou de entrada como ilustra a figura abaixo: (Goodfellow et al. 2017) (imagem).

IMAGEM

2.5. Inicialização de parâmetros

Talvez a única propriedade conhecida com completa certeza é que parâmetros precisam ”quebrar a simetria” entre as diferentes unidades. Se duas unidades internas com a mesma função de ativação, estão conectadas às mesmas entradas, então essas unidades devem ter diferentes parâmetros iniciais. Caso elas tenham os mesmos parâmetros iniciais então um algoritmo de aprendizado determinístico aplicado à um modelo e função de custo também determinísticos, irá conseguintemente atualizar essas duas unidades da mesma forma. Mesmo se o modelo ou o algoritmo de treino for capaz de usar aleatoriedade para computar diferentes atualizações para diferentes unidades (por exemplo, dropout), na maioria das vezes é melhor inicializar cada unidade para computar uma função diferente de todas as outras unidades. Isso pode ajudar a garantir que nenhum padrão de entrada seja perdido no espaço nulo da forward propagation e nenhum padrão de gradiente seja perdido no espaço nulo da back-propagation. O objetivo de ter cada unidade computando uma funçao diferente motiva a inicialização aleatória de parâmetros (Goodfellow et al. 2017).

Comumente, estabelecemos constantes os valores dos biases escolhidos de maneira heurística, e inicializamos apenas os pesos aleatoriamente (Goodfellow et al. 2017).

2.6. Word Embeddings

Word embedding é o nome coletivo de um conjunto de técnicas de modelagem de linguagem e de aprendizado de recursos no processamento de linguagem natural, em que palavras ou frases do vocabulário são mapeadas para vetores de números reais. Conceitualmente, envolve uma incorporação matemática de um espaço com uma dimensão por palavra para um espaço vetorial contínuo com uma dimensão muito menor.

2.7. Redes Neurais Recorrentes

2.7.1. (GRU)

Células recorrentes bloqueadas são definidas pelas seguintes formulas:

FORMULAS

2.7.2. Longa memória de curto termo (LSTM)

Células de longa memória de curto termo são definidas pelas seguintes formulas:

FORMULAS

3. Metodologia

3.1. Organizando dados

O banco de dados utilizou da técnica de separação de treino e teste, onde 90% dos dados foi atribuída para treino e 10% para validação. Essa técnica é conhecida como train-test-split. Em abordagens clássicas de aprendizado de máquina, normalmente é utilizada a

validação cruzada, porém em casos de aprendizagem profunda a mesma é ineficiente. Como modelos de aprendizagem profunda possuem grande volume de dados, a abordagem mais popular em relação a desempenho e resultado é escolhida nesse trabalho.

A segmentação a seguir representa como os dados foram separados:

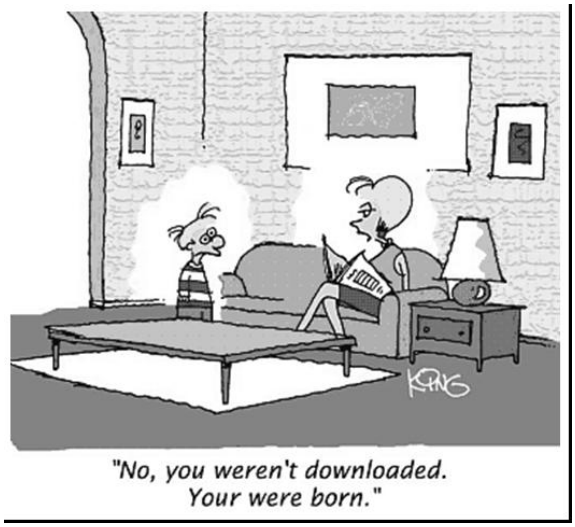

Figure 1. A typical figure

A segmentação de testes não está inclusa nos dados públicos. Tal teste é feita através de dados não divulgados na plataforma Kaggle.

3.2. Definindo modelo inicial

Um modelo inicial foi definido para obter o primeiro resultado e utilizar uma técnica bastante popular no segmento de aprendizagem profunda. A técnica de iteração, consiste na ideia de definir uma arquitetura inicial, e definir hiperparâmetros rapidamente. Como ponto de partida, foi utilizado uma arquitetura bastante popular e extremamente efetiva, sendo sem transferência de aprendizado, o estado da arte. A rede foi definida da seguinte forma:

Embedding se refere a essa codificação. Utilizamos o conceito de vetores globais para representação de palavras (GloVe) pré-treinados em um crawler global.

Dropout previne o encaixe exagerado da função aos dados, logo faz o modelo generalizar melhor.

Célula define qual componente interno da rede neural recorrente será utilizado. Nessa arquitetura utilizamos variações entre 128 e 256 nós.

Convolução utiliza uma janela deslizante unidimensional com um filtro de tamanho três. Busca encontrar características e padrões.

Pooling diminuem tamanho da saída da convolução.

Concatenção junta os tensores de saída dos poolings.

Predição faz uma rede neural totalmente conectada e utiliza a função de ativação sigmoid, sendo que retornos maior que 0.5 serão considerados verdadeiros, e menores como falso.

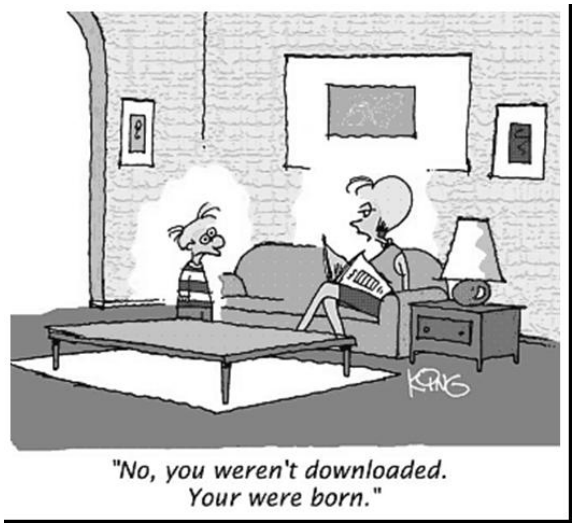

Figure 2. A typical figure

Os hiperparâmetros do otimizar foram definidos através dos valores indicados no artigo do mesmo. A taxa de aprendizado utiliza do processo de iteração, e o valor utilizado nas análises preliminares é arbitrário, sendo inicialmente 1x10-3. Os pesos das células foram inicializador através da inicialização Glorot (Bengio Glorot, 2010).

4. Análises Preliminares

Em primeira instância os modelos foram bastante satisfatórios, alcançando acurácia próxima à melhor submissão deste problema.

QUADRO

Podemos observar que nesse conjunto de dados a bidirecionalidade não trás benefícios nestas instâncias. Outro fator de relevante é o desempenho do GRU ter sobresaido em relação a LSTM. O GRU é um modelo ”simplificado” da LSTM tradicional.

Análises utilizando modelos de linguagem e transferência de aprendizado ainda estão sendo desenvolvidas.

5. Considerações Finais

Os modelos testados apresentaram um resultado satisfatório, porém não melhor que o melhor resultado atual. Técnicas como transferência de aprendizado e modelo de linguagem não foram aplicadas. Há a possibilidade de reelaborar a célula que constrói a rede neural recorrente para atender melhor a este caso. Talvez a aplicação de uma rede de capsulas (Hilton et al. 2017) proporcionaria um resultado superior devido a não variação em translações, uma das características desta arquitetura.

6. Referências Bibliográficas

Bibliographic references must be unambiguous and uniform. We recommend giving the author names references in brackets, e.g. [Knuth 1984], [Boulic and Renault 1991], and [Smith and Jones 1999].

The references must be listed using 12 point font size, with 6 points of space before each reference. The first line of each reference should not be indented, while the subsequent should be indented by 0.5 cm.

References

Boulic, R. and Renault, O. (1991). 3d hierarchies for animation. In Magnenat-Thalmann, N. and Thalmann, D., editors, *New Trends in Animation and Visualization*. John Wiley & Sons ltd.

Knuth, D. E. (1984). *The TEX Book*. Addison-Wesley, 15th edition.

Smith, A. and Jones, B. (1999). On the complexity of computing. In Smith-Jones, A. B., editor, *Advances in Computer Science*, pages 555–566. Publishing Press.