”あなたの日記を AI が読みます”

工学部航空宇宙工学科三年 野本陽平 (03-180332)

2018 年 8 月 10 日

動機

僕は毎日一日の終わりに日記を書いています。朝にはついつい占いなんかも見てしまいますが、日記は占いとは違い「起こるかもしれない未来」ではなく、「実際に起きた過去」であり「繰り返すことも有りうる未来」です。受験勉強で復習がとても大事だったように、人生全般においても日記をつけることでより良い未来を得ることができるはずです。... しかし受験勉強と日記の違いは、「日記は他人に見られたくない」ということです。(あくまで僕の場合は、ですが！)AI だったら「今日はこんな一日だった！」ということを日記を読んで助言してくれても恥ずかしくないのでは？と考えたのが今回の課題の 1 番の動機です。その第一歩として、良い一日だったかそうでもない一日だったかを教えてくれるクラスを作りました。
また、今回自分がファウンダーとなり東大の産学共創が開催する Summer Founder Program から 60 万円ほどの支援を受け、他学科の友人たち 5 人と「AI 目覚まし」を作ることになりました。AI 目覚ましについての詳しい内容はここでは触れませんが、内容は大いに「自然言語処理」が関連しています。今回の課題は自然言語処理の中の「感情分析」という領域のものなので、AI 目覚ましに繋がる内容をここで学ぶことができるのではないか？！と思ったことも動機の一部です。
さらに友人から「鳥の鳴き声を深層学習してどの鳥か判断するアプリケーションを作って欲しい」という仕事を依頼されており、より一層の機械学習の習得が望まれていました。
留学中に現在 Ph.D 過程中で東大航空卒の西成さんにお会いし、「東大航空のカリキュラムは少し古めなので、ソフトウェア関連を中心に自分で深めに勉強しておくとより良いかもね」という趣旨のアドバイスも頂戴していたことも同期の一つだと思います。長くなってしまいましたが、これら全てがこの課題に向けての僕のモチベーションです。
とはいえこちらではインテンシブな授業を履修していたり人付き合いなどもあったため、あまり課題に多くの時間を割けなかったことが悔やまれます。

解説

FinalAssignment クラスは、主に 50000 個のデータを読み込むための makingcsv 関数と打ち込んだテキストを makingcsv 関数の作る csv ファイルと比較し分析することを担当する analysis 関数でできています。サブの関数として tokenizerporter 関数では、入力したテキストの三人称単数や時制を標準に直す機能を実装しています。ですがかなり古典的なアルゴリズムを採用しているため、性能はそこまで高くありません makingcsv 関数と analysis 関数の中で利用しています。mergedictaddvalues 関数では、collections ライブラリを利用して辞書の足し算を実装しています。本課題で利用している IMDb 映画レビューセットは、自然言語処理の一分野・文章の極性を分析する意見マイニングに頻繁に用いられるデータセットです。

データセットの中には映画のレビューとそのレビューが「肯定的」か「否定的」かが入っています。(どうやら IMDb では 1 個から 10 個の星と文章の形態で映画のレビューを行なっているようで、ここでは 6 個以上の星の投稿を「肯定的」・5 個以下の星の投稿を「否定的」と捉えているようです。)
メインの makingcsv 関数は 50000 のレビューデータを読み込み、不必要なデータをクレンジングし、ポジティブな言葉・ネガティブな言葉を 300 個ずつ、出現回数と対応させた辞書の形で返します。標準的な PC だと実行には 10 分弱を要します。この際に初めの 10 個の単語は This などの感情に関係ない純粋な頻出単語であったため切り捨てました。また何故ハイパーパラメータとして 300 個を選んだのかというと、何回か実験してみたときに出現頻度が低すぎる言葉もやはり感情に関係ないことがわかったためです。すなわち「多すぎず少なすぎない感情を表す単語」が辞書の形で返されていることになります；そして最後に同じディレクトリ内に csv ファイルとして書き出しています。(書き出しているのは逐一やるには計算コストが大きすぎるため。)
analysis 関数では受け取ったテキスト (日記) を細かく噛み砕き、makingcsv で作った csv ファイルのデータと比較しています。最後に非常に大雑把な関数ではありますが、「今日の充実度」を計算し返して関数は終わります。

改善点

まず単純なことですが、映画の感想のデータセットを使ったために語彙に偏りがあります。もっと良いデータセットがあれば良いのですが...。
クレンジングに原始的な関数を使っている点も挙げられます。もし結果が不満のあるものだったらここを真っ先に直すつもりでした。計算の負荷が大きくなってしまう事情もあるので、それなりに満足いく結果が得られたためにここは直しませんでしたが、容量の大きい PC ならここは直せる余地があると思います。
入力時に現在ポジティブな気持ちかネガティブな気持ちかを入力しておけば、新しくデータセットを更新できる機能は可能なら実装したかったです (時間切れでした。)。映画ではなく日記に向けての言葉なのでより予測の精度を高めてくれるはずだと思います。
変数 prediction の算出方法を変えることは大事だと思います。直感的には log を使ってみても面白かった可能性がありますが、計算上のバグを生む可能性が高くなります。
RNN を用いてさらに精度を向上させることも考えられますが、それは夏休み中にやります。
アプリケーションの形にしようかということも考えたのですが、python はアプリに適していないのでやめました。ですが本当に使おうと思ったらアプリの形が最適のはずではあります。

アピールしたい点

実際にやってみるとそれなりに高い精度で予想してくれることがわかります！参考資料は後ろに書いてありますが、あくまで自作を意識したので殆ど引用していません。改善点を考えたことも評価していただきたいです！動機のところに書きましたように夏以降もゴリゴリプログラミングしていく！というところは評価していただけると幸いです (が、そんなことを言ったらなんでもアリになってしまうと思うのであまり期待しないでおきます。)

参考情報

「Python 機械学習プログラミング」
$https://github.com/rasbt/python-machine-learning-book-2nd-edition$
「自然言語処理 100 本ノック」
$http://www.cl.ecei.tohoku.ac.jp/nlp100/$

連絡先

$n.yohei.1717@gmail.com$