Intro_to_AI HW4

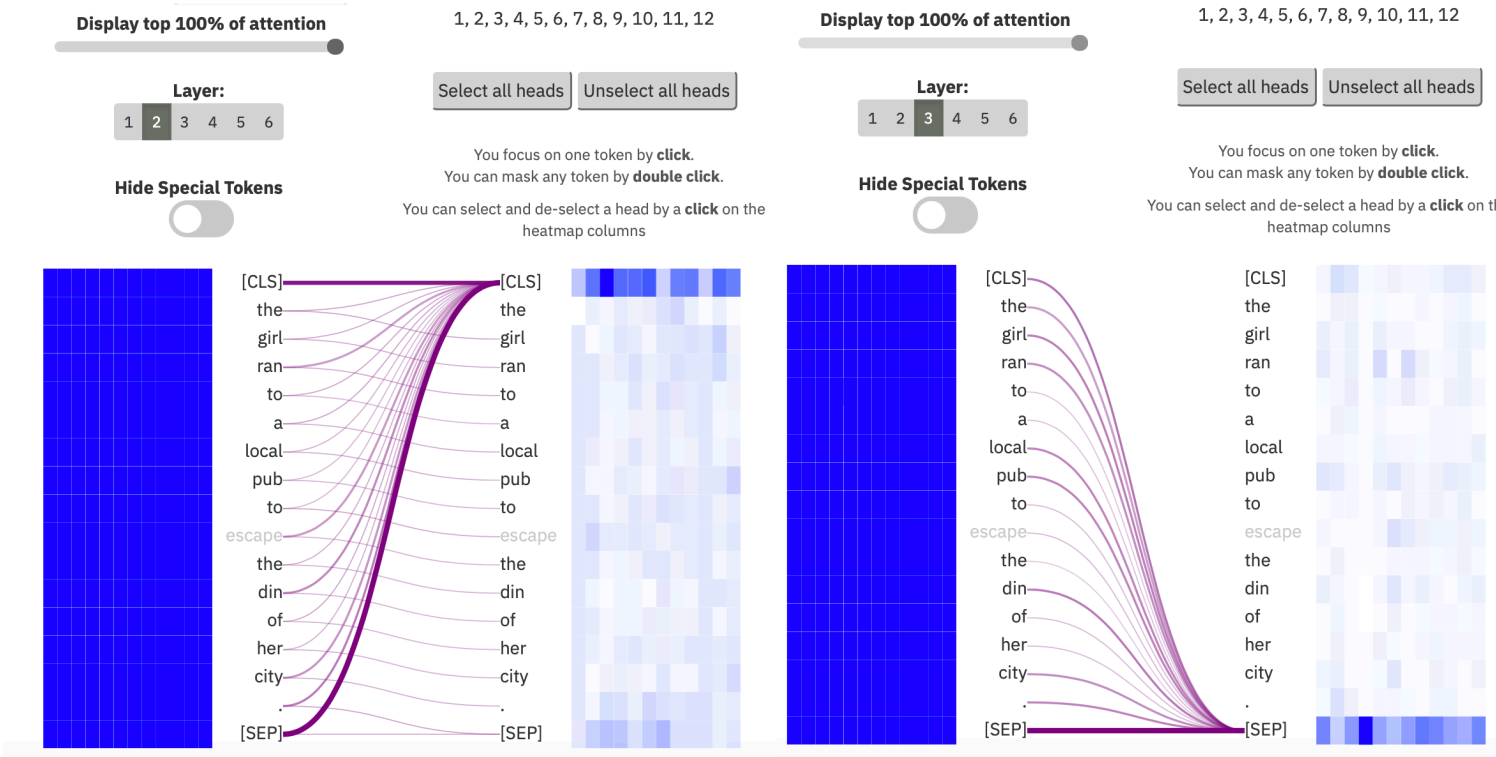
109550155 端木竣偉

Part1 exBERT

先講解筆者在能順利完成Part1要求前，所需要理解的一些前置觀念，Bert內部的layer採用是採用attention機制，意思是找出對於每個word而言，哪些word比較”重要“。而每一層之中採取Multi-head，比較抽象的解釋是，從不同的角度去觀察重要程度，最後再綜合所有結果得出結論。而我們要使用的model是distilbert-base-uncased，其中Distilbert可以說是精巧版的Bert，uncased則是表是說將句子中的字母一律轉換成小寫，並去除重音符號。

接下來將講解我在使用exBert時觀察到的結果和原因解釋，解釋的部分因為之前筆者對於Bert沒有深度瞭解，無法給出具高度可信的結論，所以研讀了不少網路文章與論文，並在exBert論文的reference中找到了具高度相關的論文(What Does BERT Look At?,Kevin Clark 2019)，從中引用了部分結論與觀察：

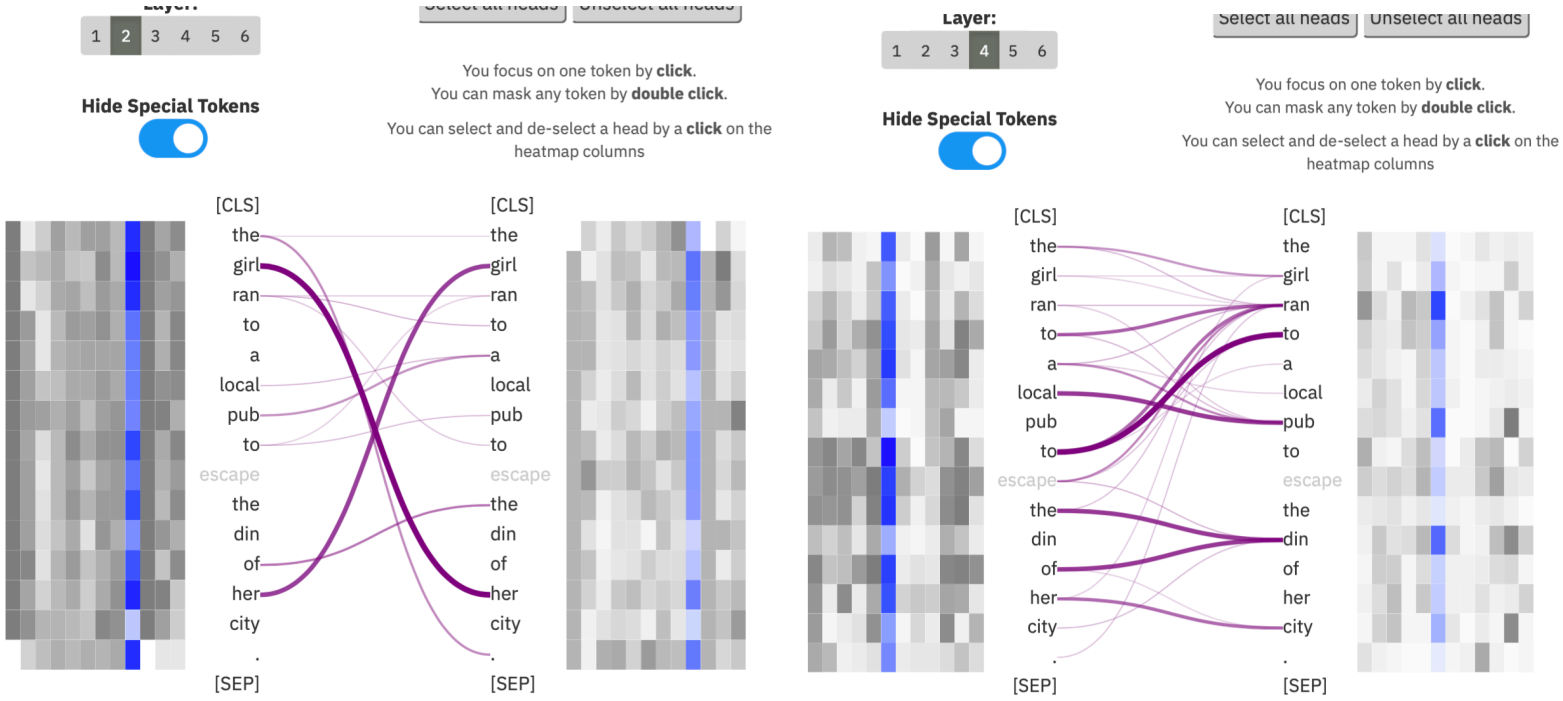
我使用的是網站自動給的例句「The girl ran to a local pub to escape the din of her city.」



我將隱藏special Tokens關掉，並顯示所有的attention，得出來的結果非常神奇，Layer2很大程度的attemtion都集中在[CLS]，至於3~5層則幾乎全部集中在[SEP]，layer6則是集中於句號上。

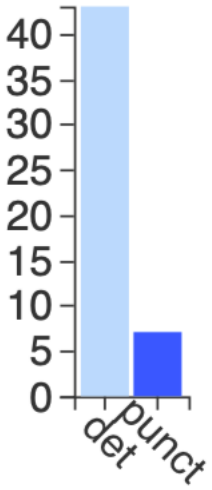
會有這樣的特別對待，可能是因為[CLS]跟[SEP]都不會被mask掉，而”.“是比”The”更加常見的詞，或句點作為所有句子最後結尾有某些特別意義，Distilbert才會特別對待。[SEP]在其中顯然最為特別，大半部分的Layer都集中於他身上，一種可能的解釋是因為前面所有的資訊都aggregate在他身上，但這樣的猜測會有一個矛盾，如果這個說法是對的，那[SEP]本身的attention將分散到所有單字上，但事實是他也集中在自己身上。另一個更可靠的說法是，對於Bert來說，將attention放在[SEP]上相當於no-op，上述的論文使用gradient-based measures of feature importance去檢驗這個假說，他們發現在[SEP]的attention集中的layers，[SEP]的gradients會變得相當小，意思是對[SEP]關注程度的大小並不會對BERT的結果有特別顯著的影響，如同no-op一般。

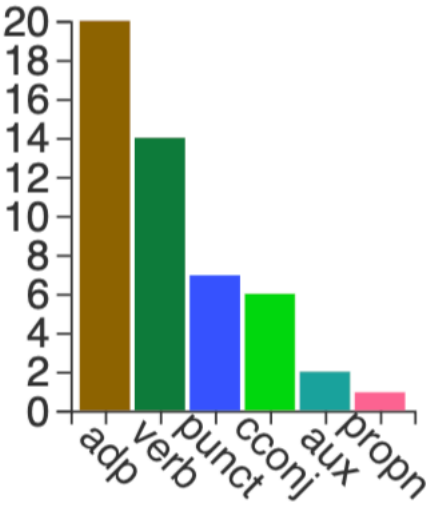
接下來的觀察就隱藏special tokens，我發現layer1、layer2分別重點關注前一個單字和後一個單字，觀察每個Head則會發現大部分都在觀察鄰近的幾個單字。我接著去觀察每一層以尋找表現特殊的head，其中我發現很特別的2-9head，girl跟her相互特別關注，代表Bert發現這兩者有特別緊密的關係，至於4-6更為特別，我觀察了一些單字發現他似乎理解文法的概念，例如girl除了連到自己之外還有run，local連接到pub，同詞性的ran跟escape除了互相連接到之外，還接到受詞pub還有din，相當厲害(下圖是2-9跟4-6的head)。



接下來我試著mask掉一些單字然後searching by embedding去預測他的詞性：

首先我選擇了escape，底下是二三層的結果，我們可以看到，在第二層的時候Bert匡出的都是the或是.這種不相干的詞，在第三層中，雖然大多數都是錯的adp，但有往正確答案(verb)靠攏的趨勢。

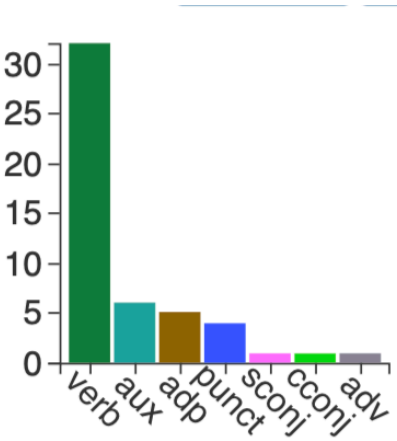


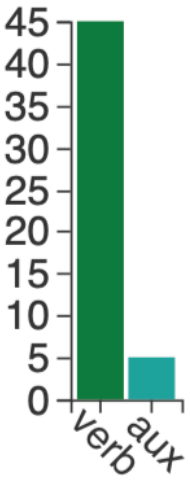


底下是四五層的結果，我們可以看到在第五層已經完全可以確定這個mask掉的word(escape)是動詞了，匡出的詞也是give、kill等動詞，而在第六層也是相同結果就不展示。

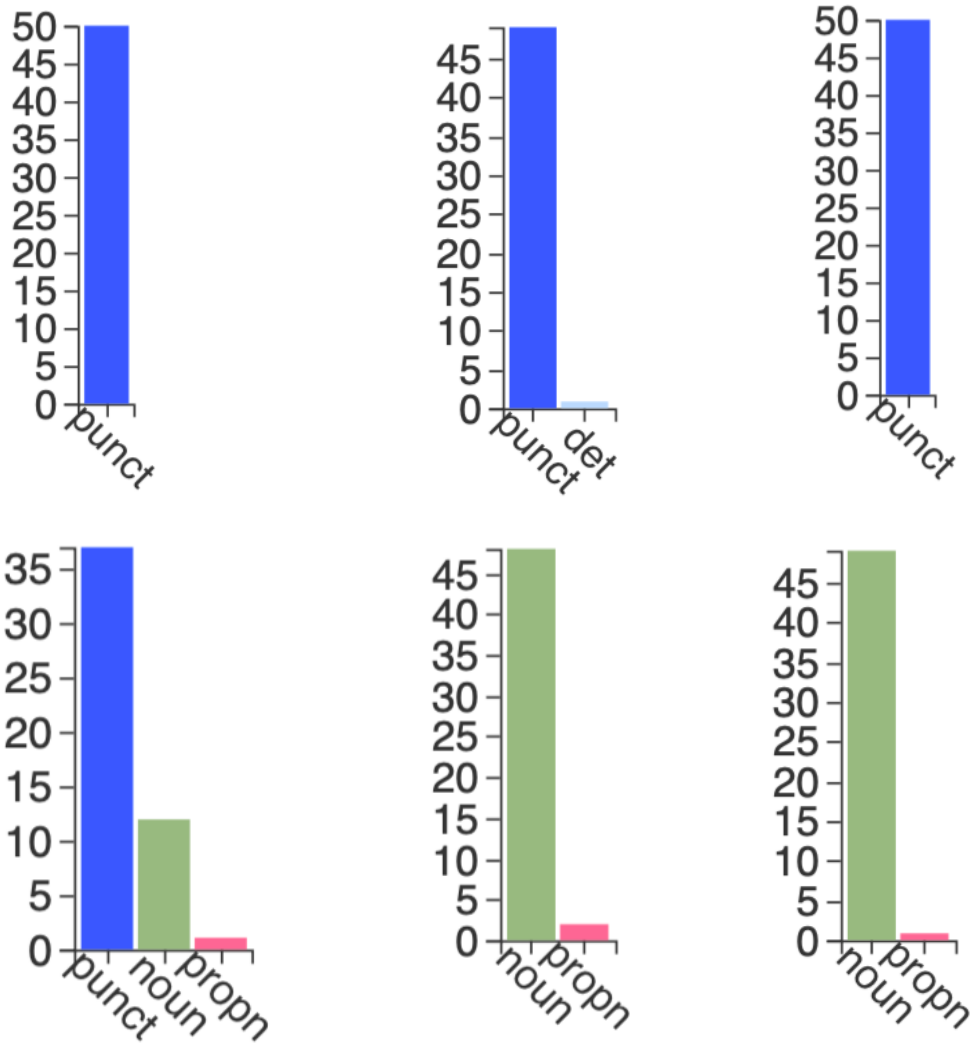
[CLS] " it would be as easy as to give the scare ##crow brains . " [SEP]

[CLS] i am going to oz to get my brains at last . [SEP]





底下是我mask掉city讓Bert去猜測詞性，於layer1~6的結果：



他也仍然找出了答案，從這裡我們就可以看出無監督學習的DistilBert強大之處。

最後我將每個字都mask掉並去觀察模型的預測狀況。

首先，「the」、「to」、「a」、「of」這類詞從文法上來說都有跡可循，而模型也大多都能判斷出來，唯一判斷錯誤的是run to的「to」，他判斷成「into」，可能是因為「run into」這個常用說法讓模型判斷錯誤，但這個意思(碰到)放在句子中顯然是不合理的，也能看出模型顯然不理解這段話的意義。

另外我mask掉local時，機率最高的是nerby再來才是local，這兩個詞意思確實有些相同。動詞的部分，「run」判斷成「goes」，我想了想覺得goes對人類也算是合理的答案，畢竟一般情況下不會用run這個詞，當然，也可以歸咎於這句話也是有些文學抽象意境，所以機器很難理解，至於escape則是意外地能預測出來。

至於「girl」則是完美預測，表示機器能理解前後文關係，從her判斷出girl，但「her」則判斷成the，一方面這樣翻譯也通順，另一方面應該是英文大部分句子都會使用the造成機器對這類用法都會預測成the。至於city就完全預測不出來，對於人來說同樣難以預測，所以無法預測相當合理。

所遇問題：

1.底子不行，毫無頭緒：

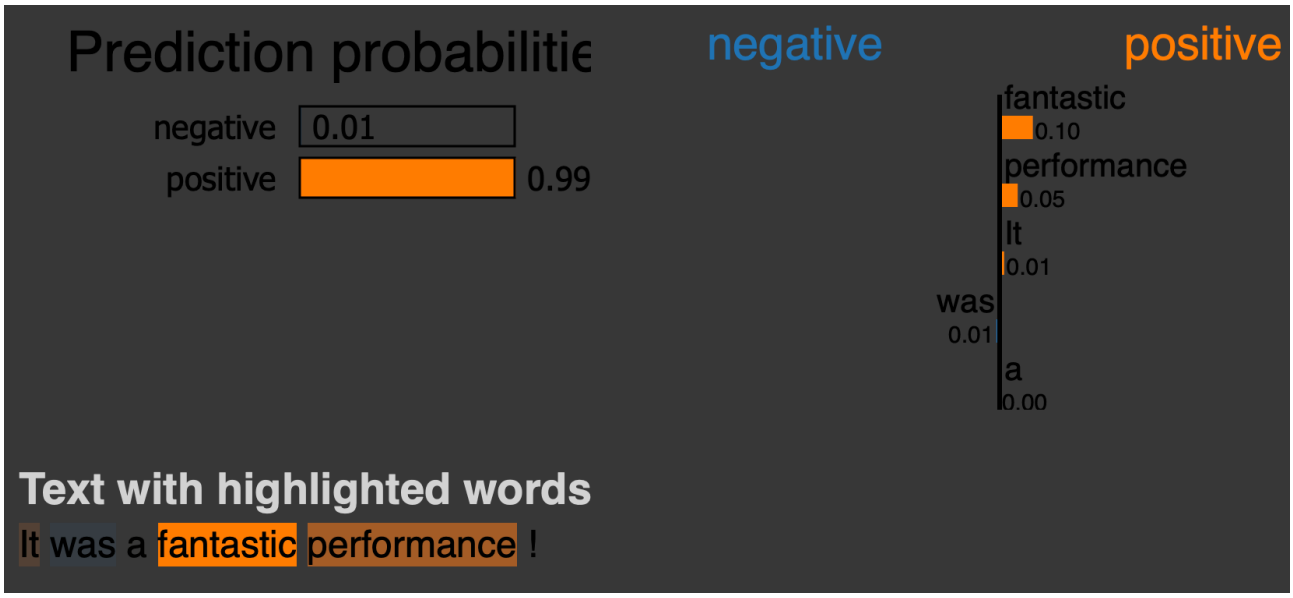
由於筆者在HW2時第一次接觸NLP跟Bert，上完課也看過李教授的課程後以為自己對Bert有一定程度的理解了，但在使用exBert的時候，完全不知道該如何解釋那奇怪的二分圖，我便先放一邊，去補強NLP的基礎(因為最近覺得這領域很有趣)，在看完LeeMeng先生的NLP文章後，發現是自己對Transformer不熟悉，才只能略懂Bert，便花時間去看了關於Transformer的文章，重點學習self-attention跟multi-head的概念後，就大概理解模型的概念了。

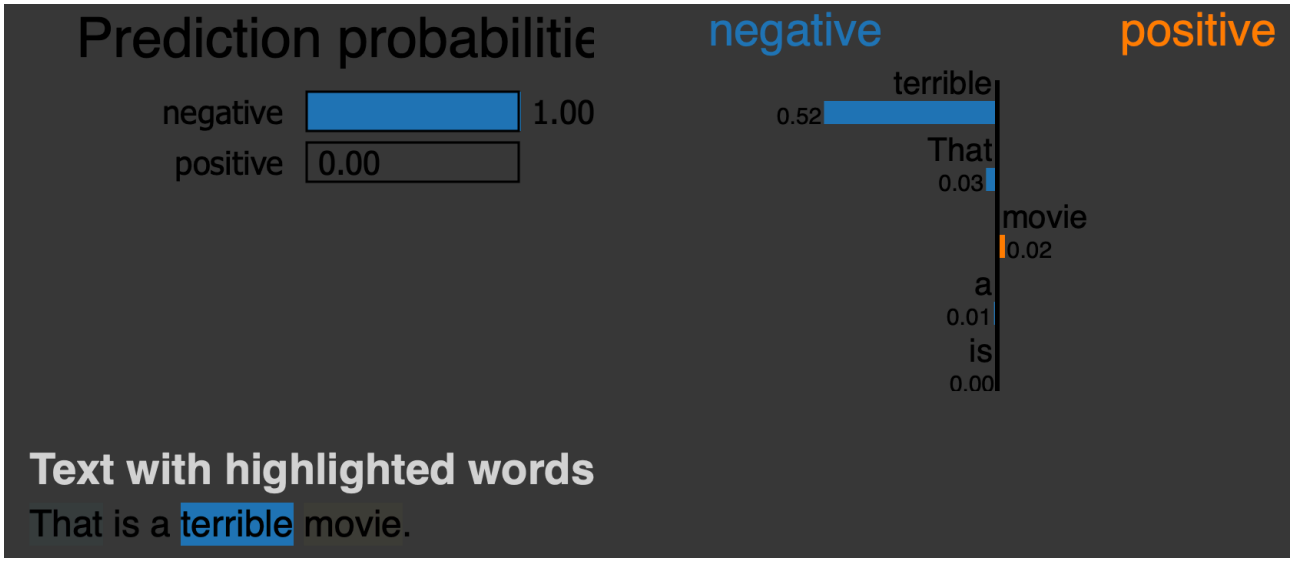
可是同樣還是不懂該如何說明為何模型會如此呈現attention的原因，在網路上查不到中文對exBert的介紹跟說明，英文資料也沒找到很清楚的，而且也沒找到針對Part1 case的資料(大概也不太可能)，所以我就決定直接看exBert網站的論文和demo影片，論文的部分雖然沒有給我太多靈感(頂多再幫我釐清Bert的概念)，但裡面提到的另一篇論文，即最開頭提過的，針對Bert進行了優秀的分析和嘗試說明，我便從中學到了不少知識跟解釋的方法，而影片的部分則教我如何使用這網站的許多功能，不然先前除了select head之外，其他的使用方法我都不會，都是從影片當中學到的，相當感謝。

接下來，我會使用自己覺得比較易懂的LIME去比較TA提供的兩個models，並分析四個句子。

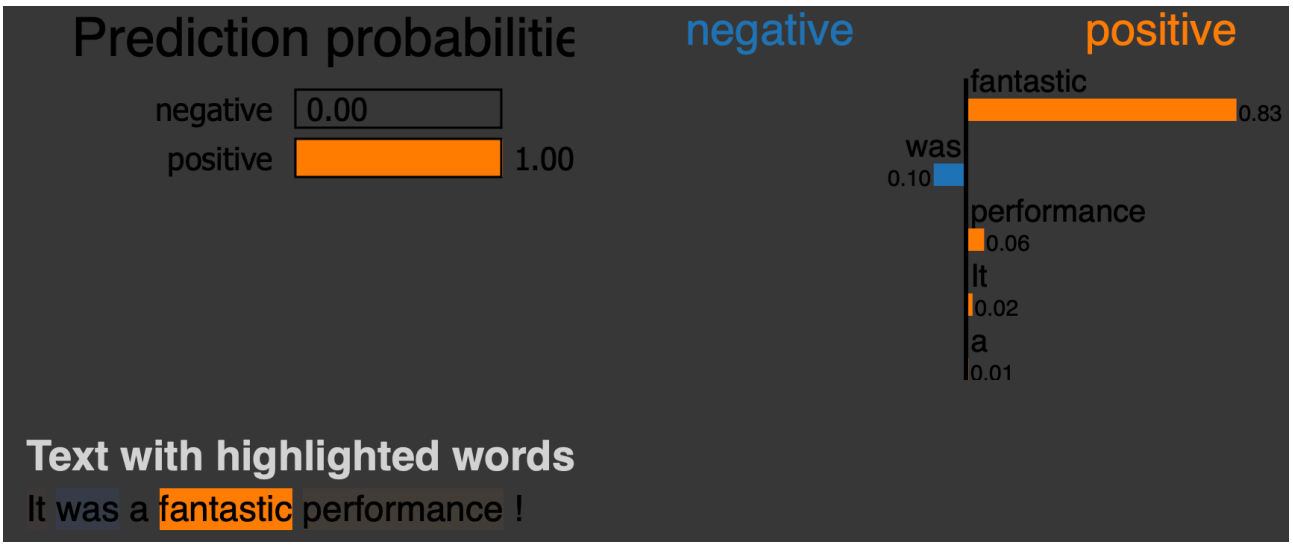
Part2 Compare two models by LIME

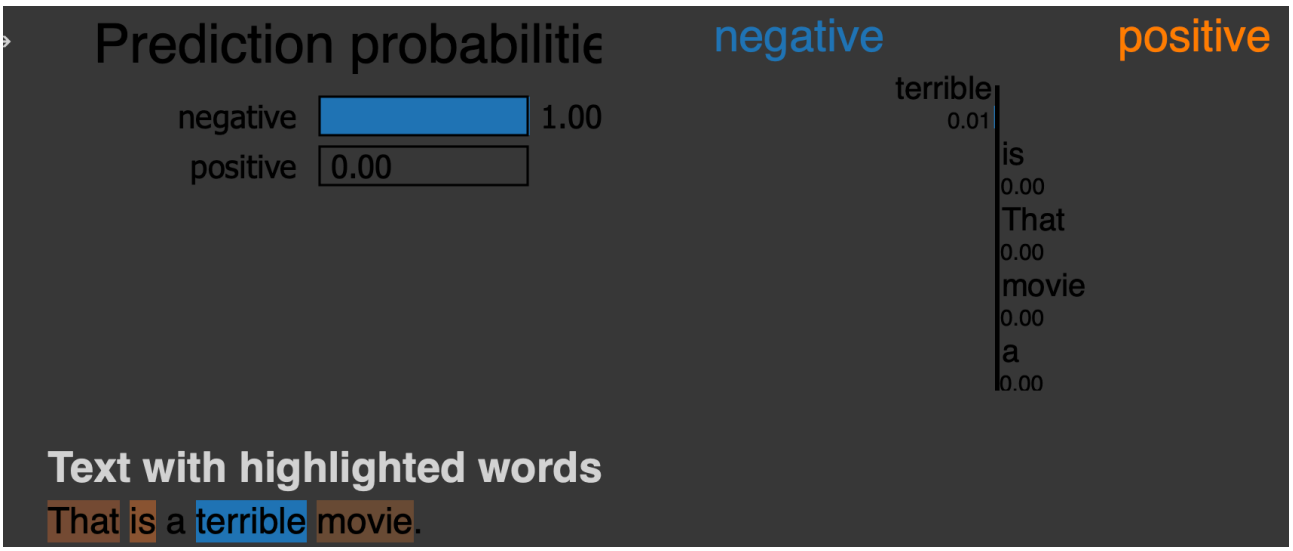
首先我使用的是範例給的兩句話：
底下是model_1的結果：





接下來是我使用model_2的結果：





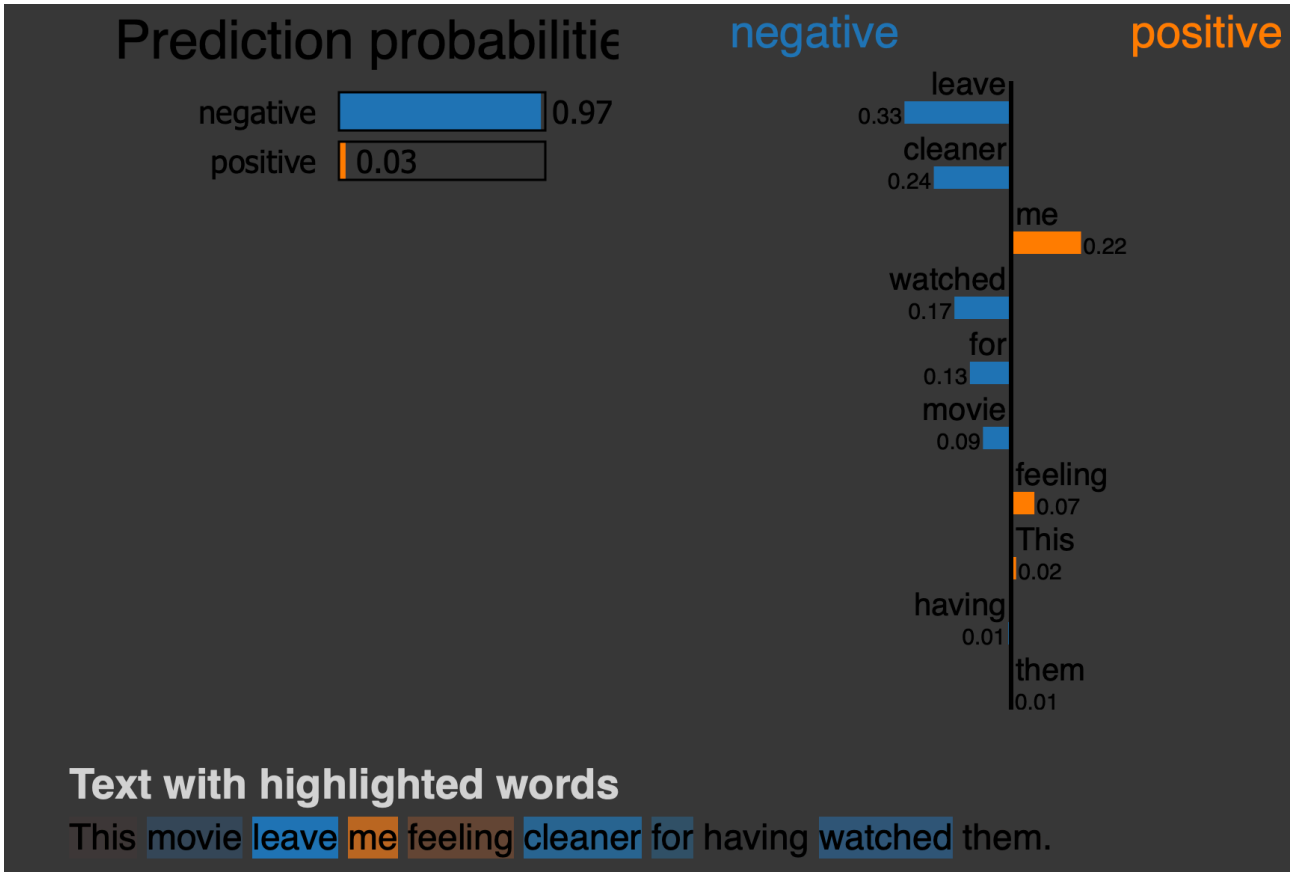
我們先看第一句話，是正面評價，而兩個模型都明明確的判斷出來，fantastic為其中最強烈的正面詞是很符合常理的判斷，但兩者都同時將performance、it視為正面詞是比較意外的結果，畢竟他們應該都是屬於沒有特別正負面意義的詞，可能的原因是兩者在dataset中比較常跟正面句子一起出現，至於在第二個模型中，將「was」視為負面詞是令人比較意外的，雖然沒有影響到這句話的判斷，但或許會成為被攻擊的因素之一。

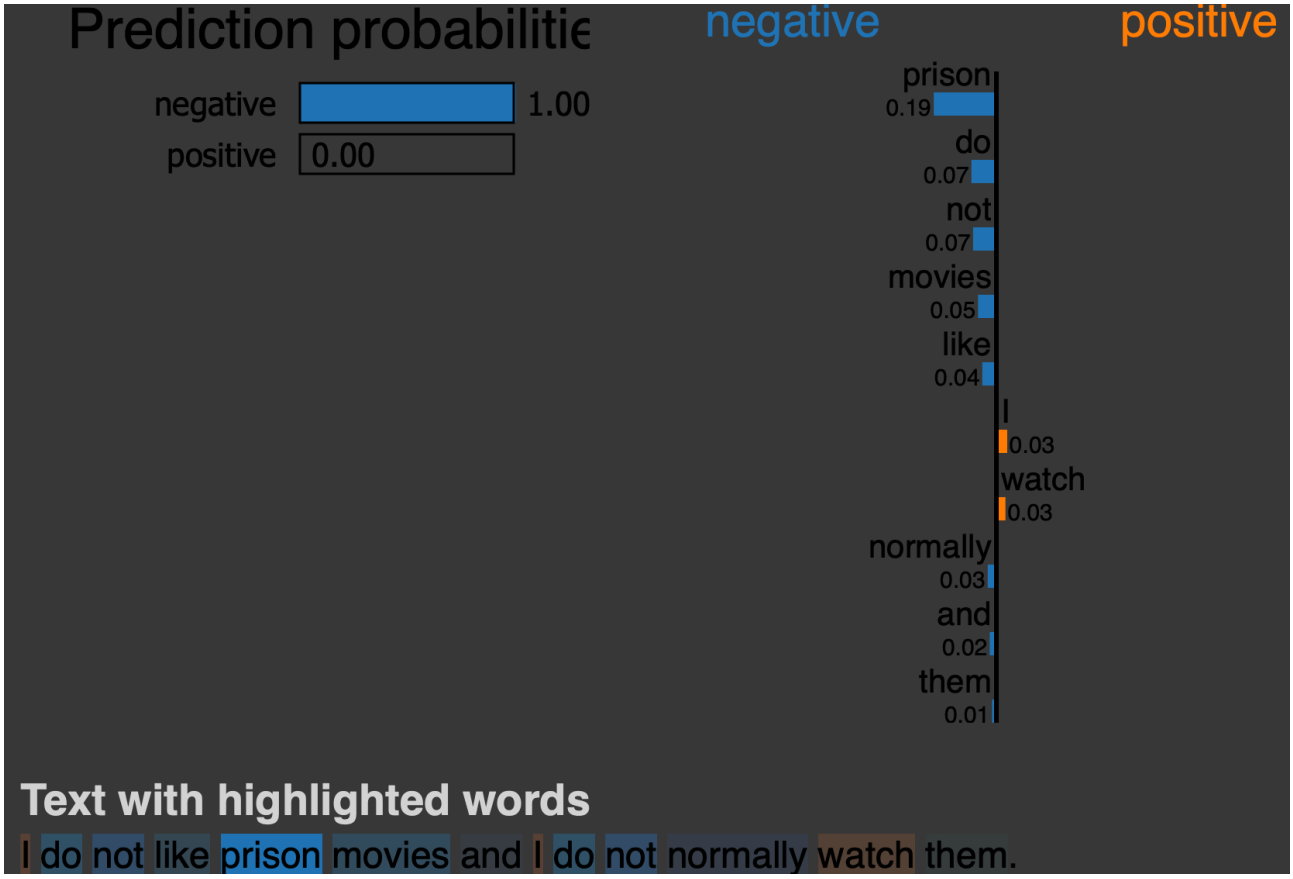
第二句話同樣也正確判斷出是負面，而terrible也是兩者判斷的關鍵因素，第一個模型的解讀很正常，給terrible很高的負值，再給一兩個不太相干的詞些許正負值，但第二個模型很奇怪的只給出了0.01的負值，其他詞則都是

0，雖然是判斷出了結果，但這樣的「解讀」實屬有些奇怪，我也沒修改任何程式碼，我認為較大的可能是LIME的解讀不佳造成的，不然這句話其實和前一句結構差不多，問題在於模型判斷的可能比較小。

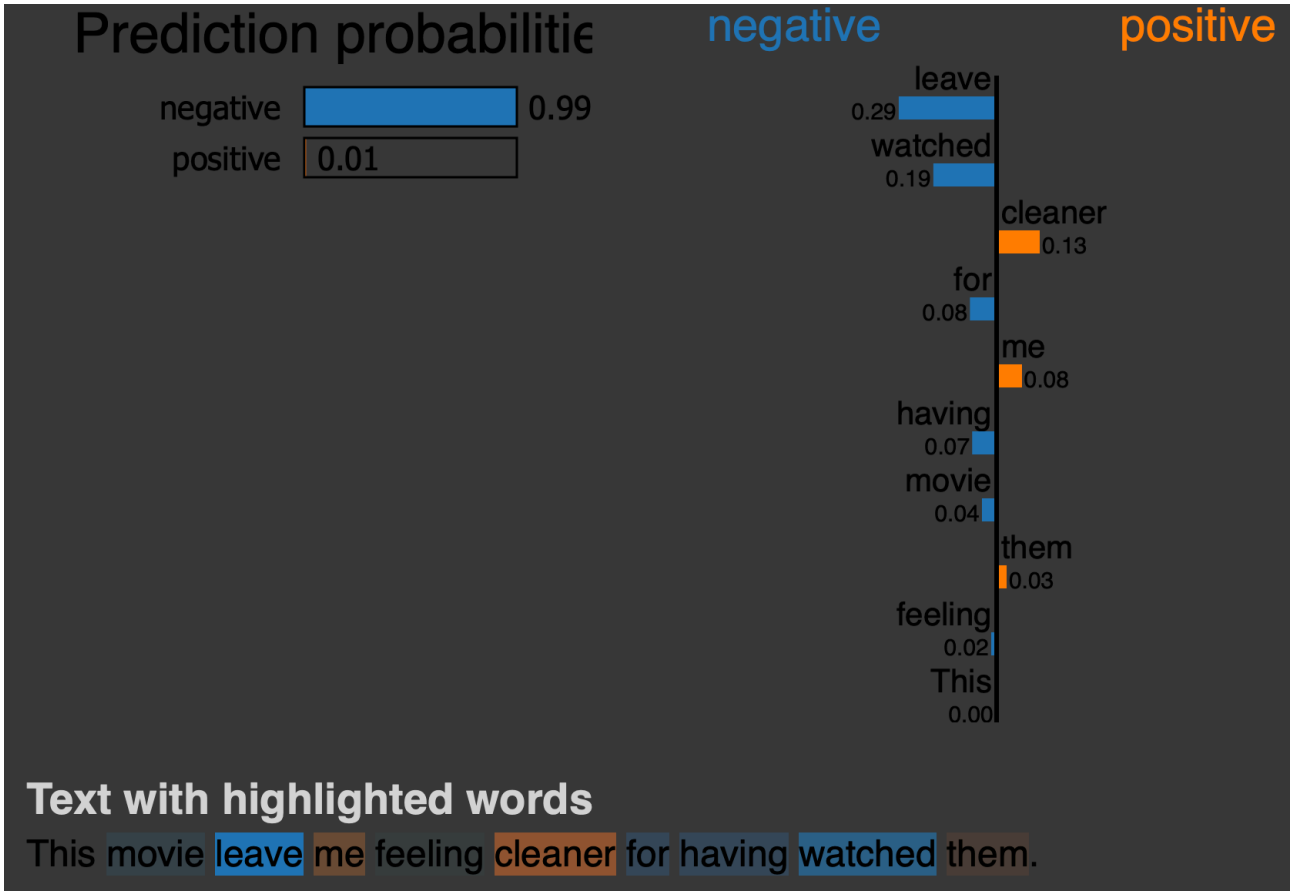
再來我選擇了自己相當喜歡且於IMDB排名第一的電影「刺激1995」在IMDB找到的兩句評論。
分別是：
正評：This movie leave me feeling cleaner for having watched them.
負評：I do not like prison movies and I do not normally watch them.

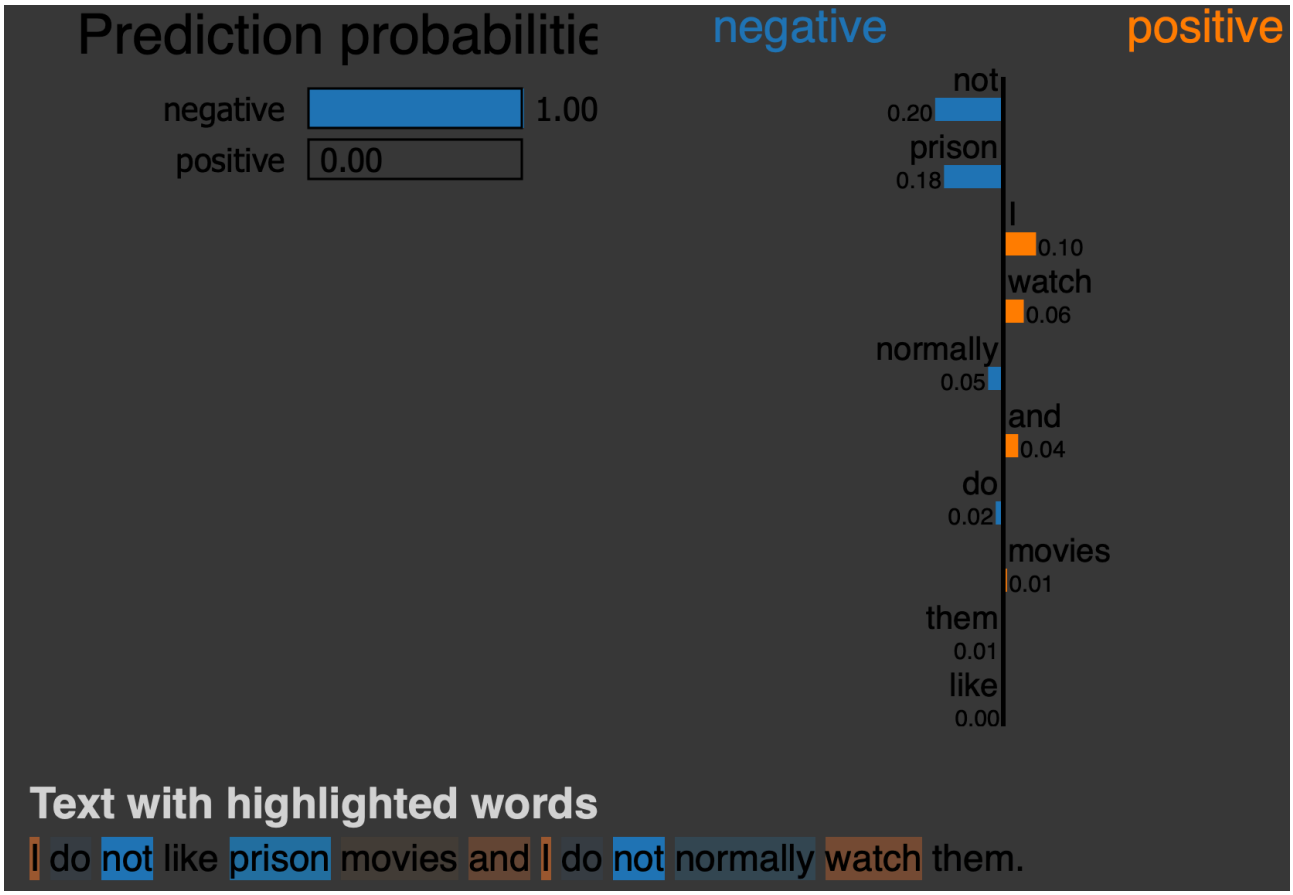
那接下來先展示model_1的結果：




Prediction probabilitie
negative
positive
negative 1.00
positive 0.00
negative
positive
prison 0.19
do 0.07
not 0.07
movies 0.05
like 0.04
I 0.03
watch 0.03
normally 0.03
and 0.02
them 0.01
Text with highlighted words
I do not like prison movies and I do not normally watch them.

再來是model_2：


Prediction probabilitie
negative
positive
negative 0.99
positive 0.01
negative
positive
leave 0.29
watched 0.19
cleaner 0.13
for 0.08
me 0.08
having 0.07
movie 0.04
them 0.03
feeling 0.02
This 0.00
Text with highlighted words
This movie leave me feeling cleaner for having watched them.



非常不幸的，第一句話並沒有預測正確，但我們同樣可以分析並從中了解如何改進模型，我們從model_1開始分析，我們先相信LIME的解釋是符合模型的(不然整個part就會毫無意義)，他將leave視為最關鍵的負面詞，雖然對人類來說，雖然在這句話沒有正負意義，但對於人類而言，leave確實是隱含負面的情緒，我覺得最大問題在於cleaner，理論上應該是富含正面意義的，但卻同樣判斷為負面意義，這應該是最大的問題，而me作為正面意義也是有些奇怪。

而在model_2上，比起model_1好的地方是，cleaner是被判斷成正面詞，但卻將watched被判斷為負面詞。我們可以得出相同的結論是，兩者都認為leave是很強烈的負面詞，而cleaner則被視為比較微弱的正面詞甚至是負面，導致判斷錯誤。

接下來分析第二句話，在兩個模型中「I」、「watch」都有些許正面的含義，如果只看這兩個字的話，會去看某部電影，那至少觀看之前對這部電影是抱有正面感覺的，不然連看都不想看了。而其他部分都是負面涵義(除了model_2的and，但and在情緒分析上不太重要就忽略)，首先，prison在兩者

都被認為是很強烈的負面詞，或許prison從常理上不是什麼正面詞，但我自己覺得這並不是很合理的判斷，畢竟市面上有不少關於監獄(prison)的電影，這樣可能會造成判斷錯誤。另外兩者也將not視為負面詞。至於其他部分，model_1將movies、like視為負面詞令我有些意外，畢竟先前句子出現movie是些微好評，like聽起來滿正向的(或許有考慮到句子前面的not?)，而在model_2的表現就比較正常。

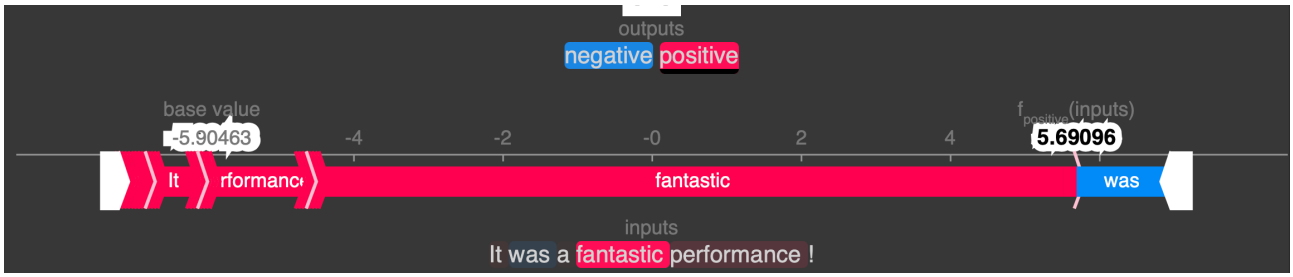
結論是，兩個模型表面上判斷的結果都一樣，但在這四個句子裡，我認為model_2的判斷詞的正負面上更為精確一些，但不管對錯，兩者判斷的趨勢是差不多的，另外我還觀察到一個特別的現象，就是兩者面對them、it、me這類詞都會給出比較正面的評價。

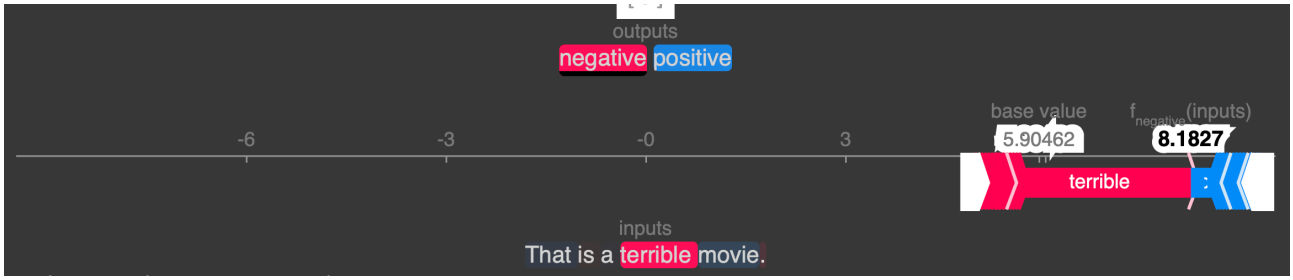
我也同樣有將SHAP的結果跑完並截圖，原本想要分別使用LIME跟SHAP去比較兩個model的差異，但我仔細分析後認為除了一些小細節外，用SHAP來分析兩者差異並不會得到與用LIME分析不同的結論，而那些差異的小細節，大多來自於SHAP跟LIME的差別，跟model無關，便選擇個人認為比較直觀的LIME去比較。

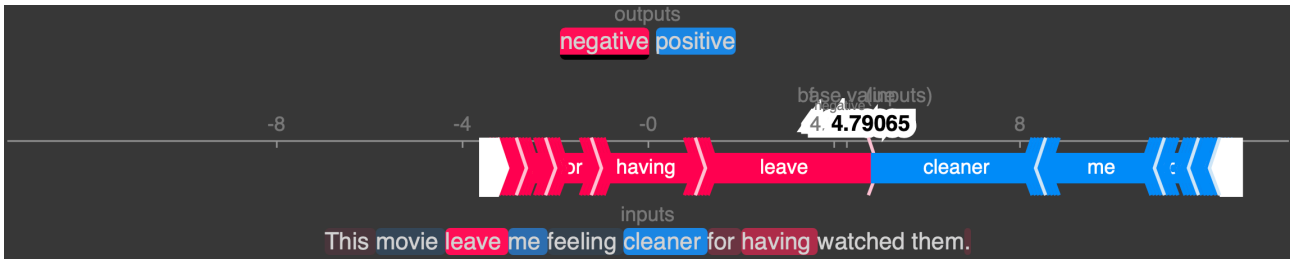
雖然Model_2在第二句話時，LIME給出了奇怪的解釋，但我稍微觀察過SHAP的解釋後發現其實也沒有太奇怪，而前面也提過，我認為model_2的表現比較合理，因此下一part就使用model_2來比較SHAP跟LIME。

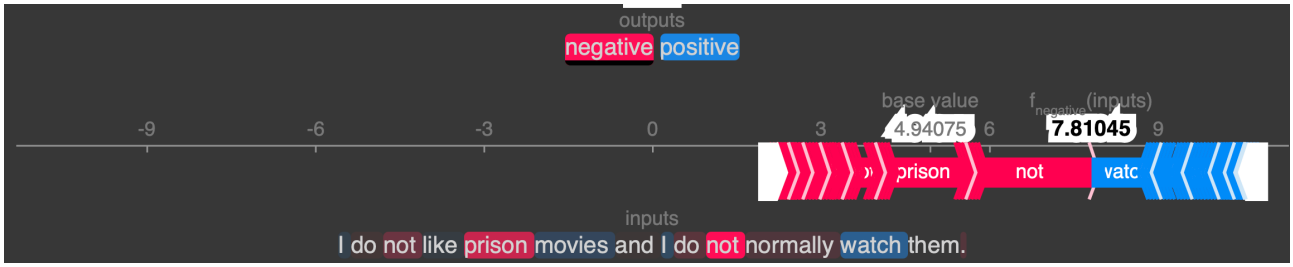
Part3 Compare SHAP and LIME

首先因為我們在這次作業中，SHAP是以local的方式解釋的，所以大致上的解釋其實和LIME差不多，並沒有出現某些LIME認為的正面詞在SHAP就變成負面的情況。









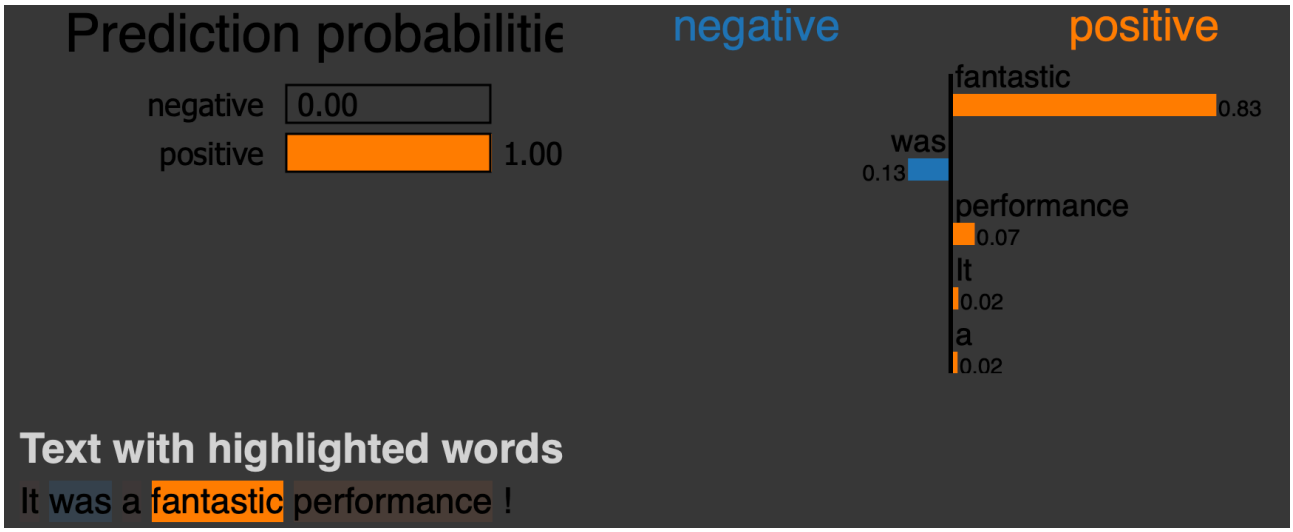
但經過一些觀察和論文查閱，我給出了以下幾點差異：

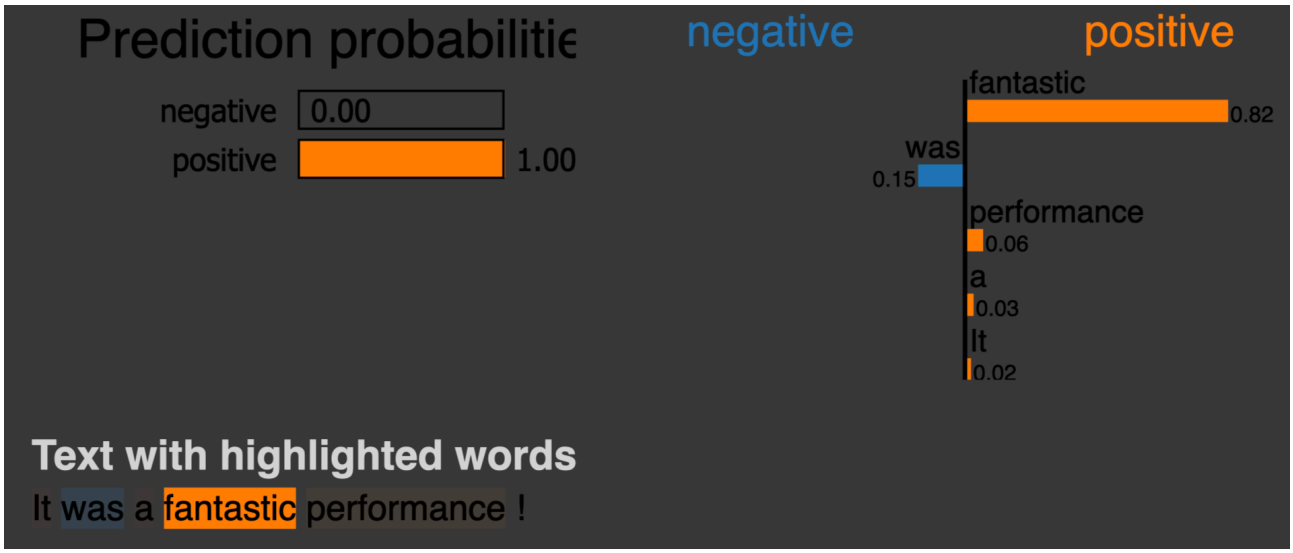
1.易讀性

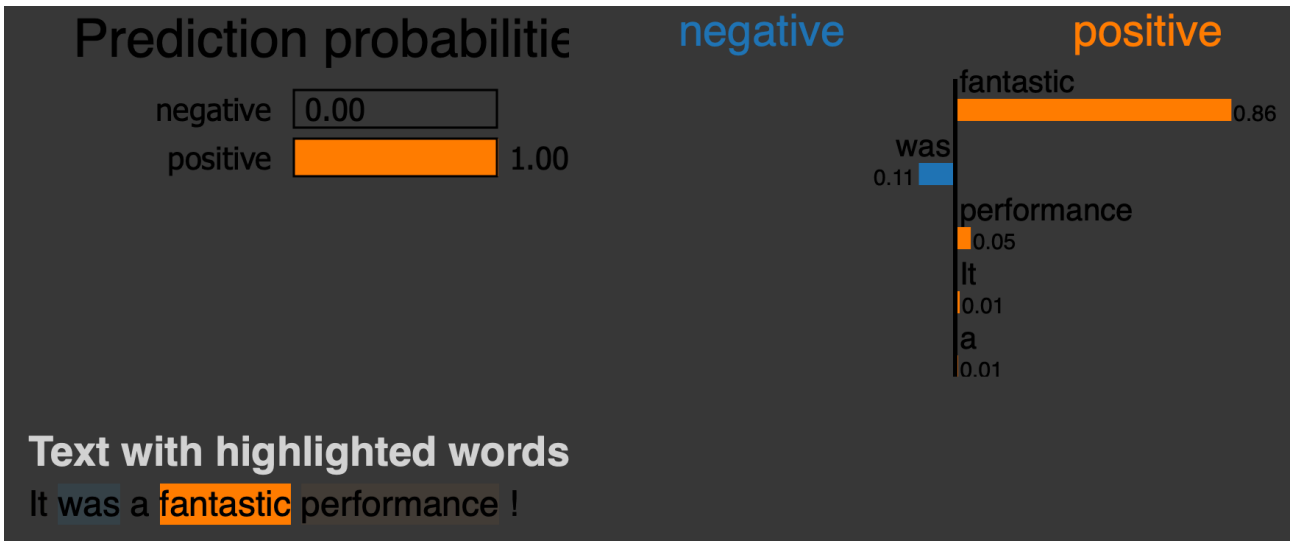
先從視覺上的來比較，我認為LIME對於不懂AI的使用者來說，解釋的呈現方式比SHAP直觀，使用者能清楚的知道1.判斷的確信度2.每個單字對於判斷的影響，而這兩個點就是explainable AI的價值所在(至少於LIME的paper裡是這麼說的)，而SHAP雖然第一眼看上去能知道判斷大致走向，但比較難閱讀細節，而且SHAP是將跟答案有正相關的詞標成紅色，負的標成藍色，第一次看的時候以為紅色代表正向詞，藍色代表負向詞而錯誤解讀，我認為一般使用者在第一次看到SHAP時也很有可能理解錯誤(不過我查閱其他SHAP的使用沒看到有這樣子的問題，可能是顯示方法不太一樣吧)。

2.穩定性

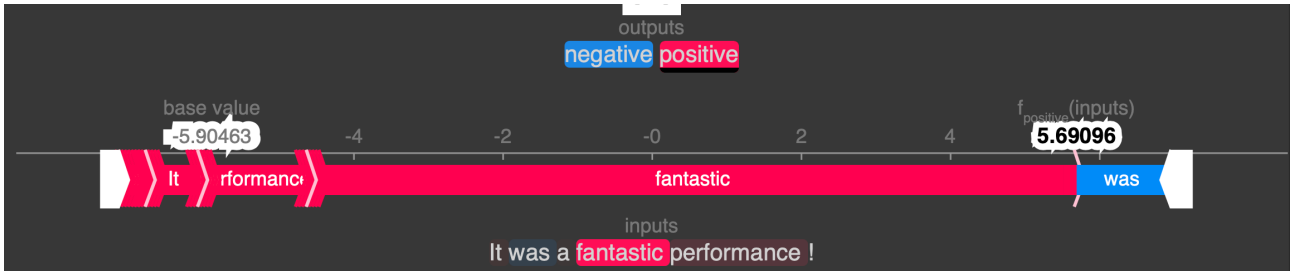
我選相同句子用LIME進行三次操作的結果如下：







我們可以從上面的數據發現，僅僅是三次測試就能發現每個字的數值都有變化，這也能側面表現出LIME的問題–不夠穩定。很可能我們經過幾次相同的操作之後，就給出差異不小的解釋，那對於使用者來說就會對於這個判斷結果有所疑慮，究竟該相信哪一次的結果才好？



那這是SHAP的結果，不管操作幾次都是固定的，使用者也會比較有信心去相信這個結果。

3.可信度

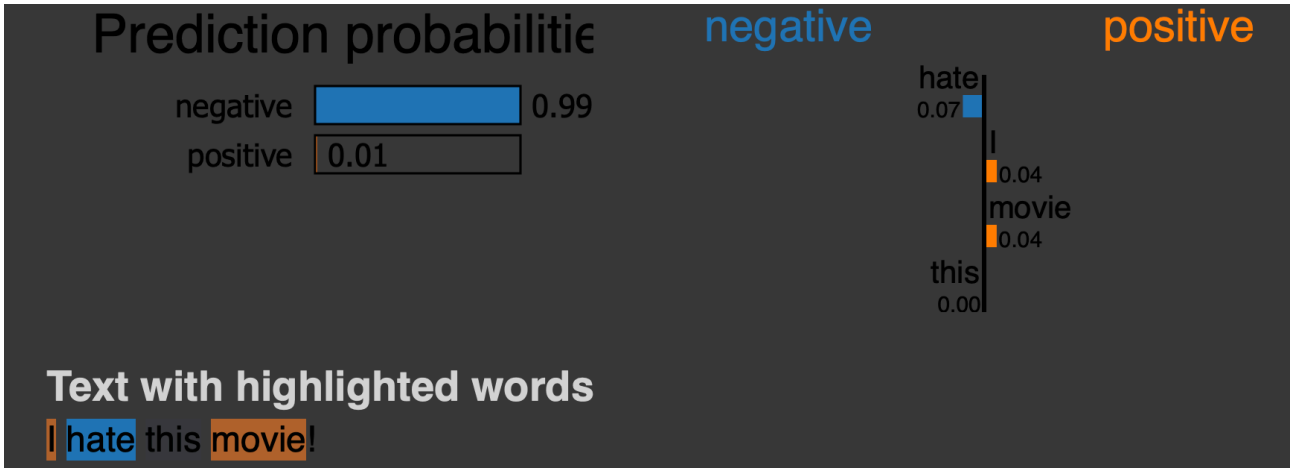
如果實際去思考LIME跟SHAP背後的運作原理，會發現LIME只是簡單的使用LR模型去進行擬合，但SHAP卻是在擁有非常堅實的理論基礎的情況下去計算出結果的，那對於使用者來說當然會偏好相信更加可信的解釋模型。

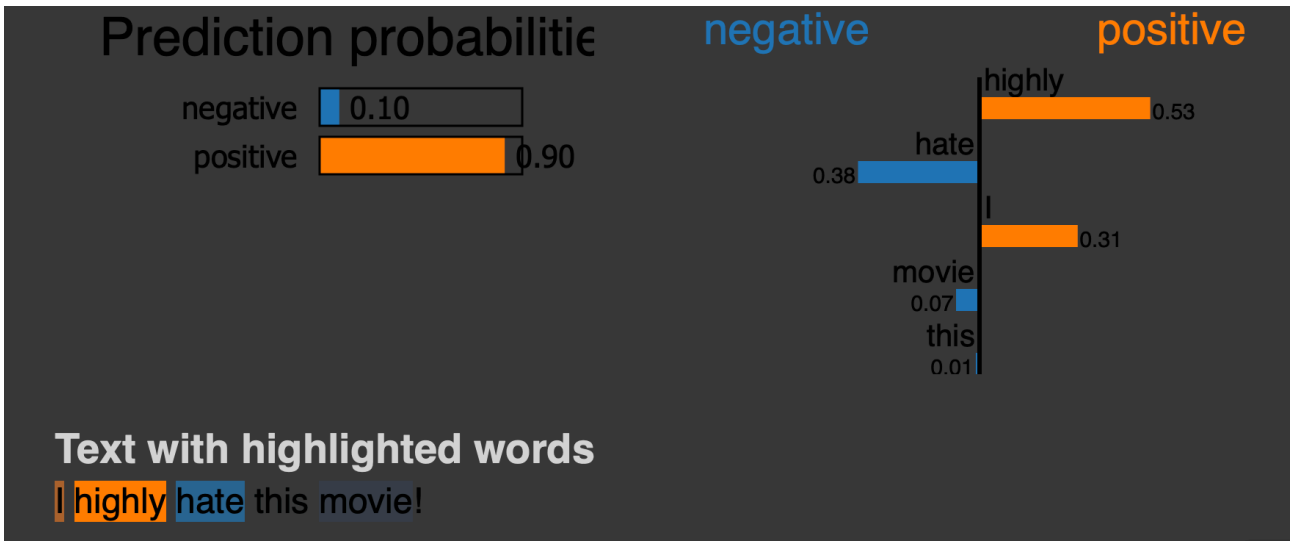
4.訓練時間差異

雖然在這次作業看不出來，但SHAP的計算複雜度相當高，是屬於指數級別的，而LIME相對就比較快速，跟上一點應是屬於trade–off的關係吧。

Attack&How to Defense

底下我給出三個例子，並討論如何防禦這類攻擊
第一個例子：word insertion

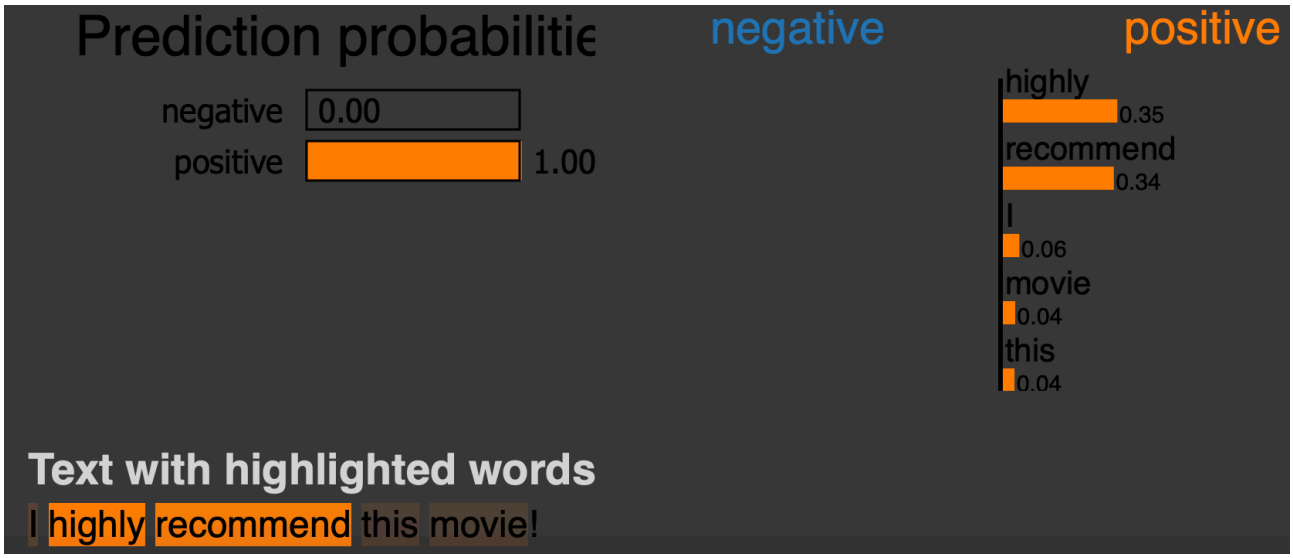


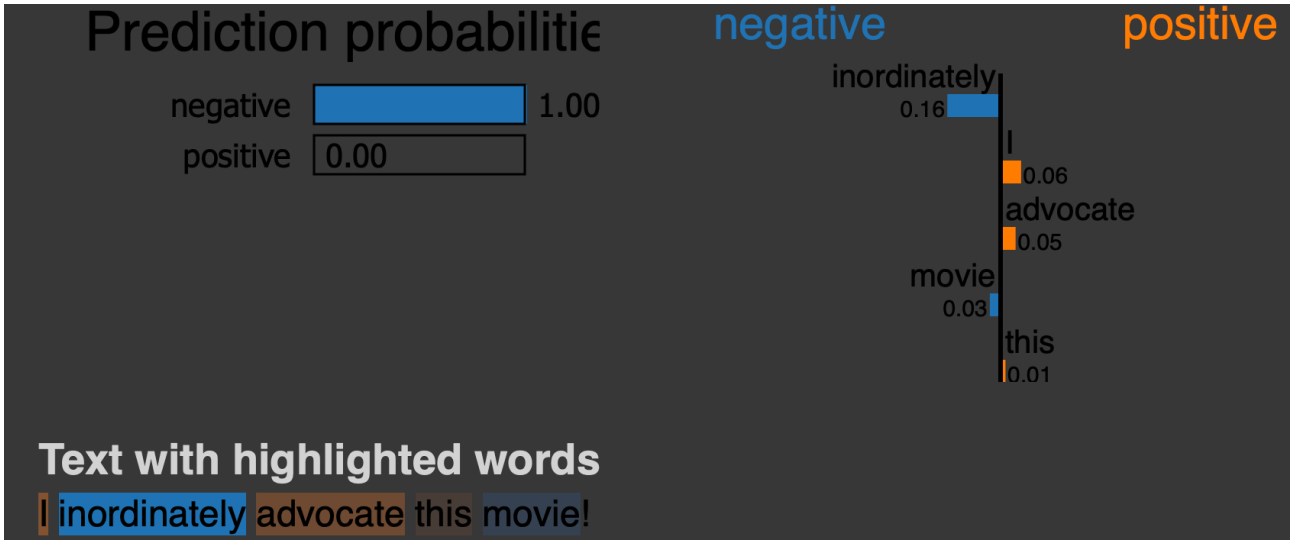


我將I hate this movie!改成I highly hate this movie就成功讓模型判斷錯誤了。
會想到這樣操作實屬偶然，因為一開始我在測試的其實是
「I highly recommend this movie.」
原本想要透過同義字替換recommend去攻擊但一直沒有成功，分析了一下結果發現正面因素幾乎都來自「highly」，因為這個詞嚴格意義上來說並沒有正

負，所以我就使用一個負面評論加上highy，模型就將同樣負面情緒的評論判定成正面了。

第二個例子：Word substitution by synonym



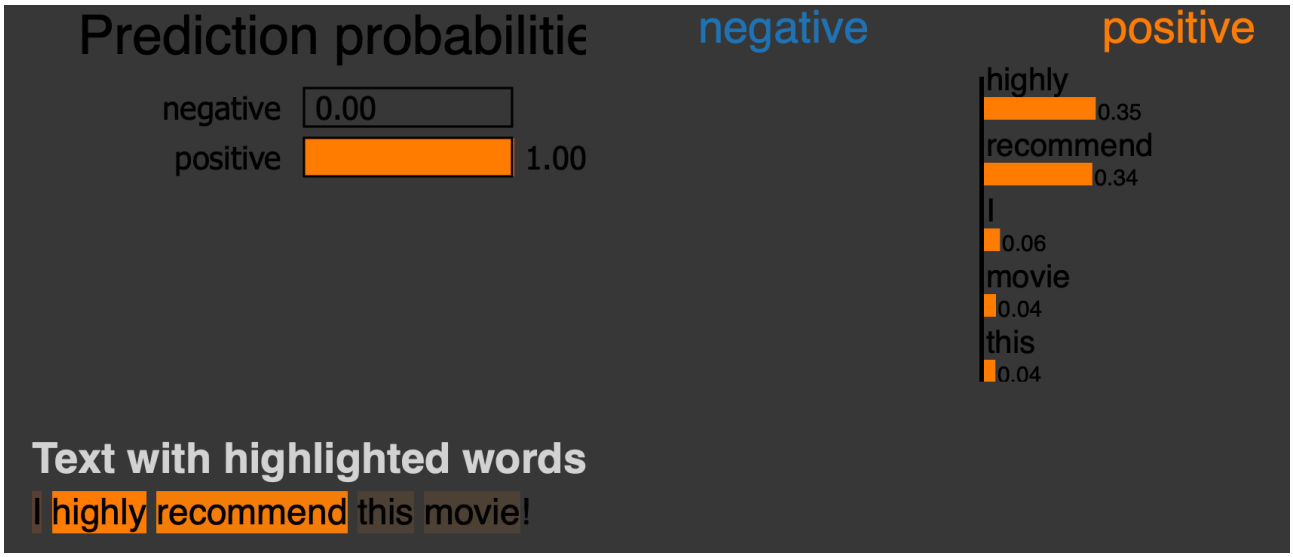


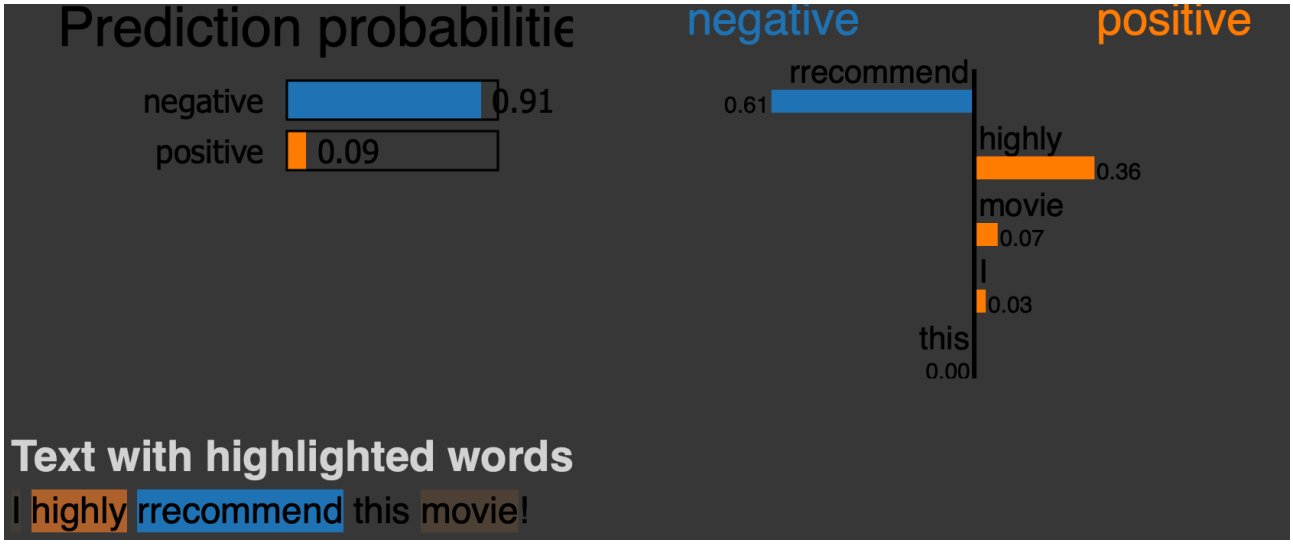
這個例子中我將highly替換成inordinately，recommend換成advocate，然後完成攻擊。

那這個例子其實是來自於我看李宏毅教授的助教講解NLP的對抗式攻擊中提到的例子，一開始只有說到替換recommend的例子，但我自己測了之後發現都沒有成功，原因上面也提到過了，就是highly的正面屬性太強，寫到這邊，筆者又去測試了如果只替換掉highly會不會就成功了，但並沒有，所以唯一的攻擊辦法就是至少替換掉這兩個字。

至於說這樣攻擊成功的原因，我認為是這兩個字不是常見單字(筆者只在考大學背七千單字時看到過幾次)，因此造成模型判斷錯誤。

第三個例子：Character level transformation



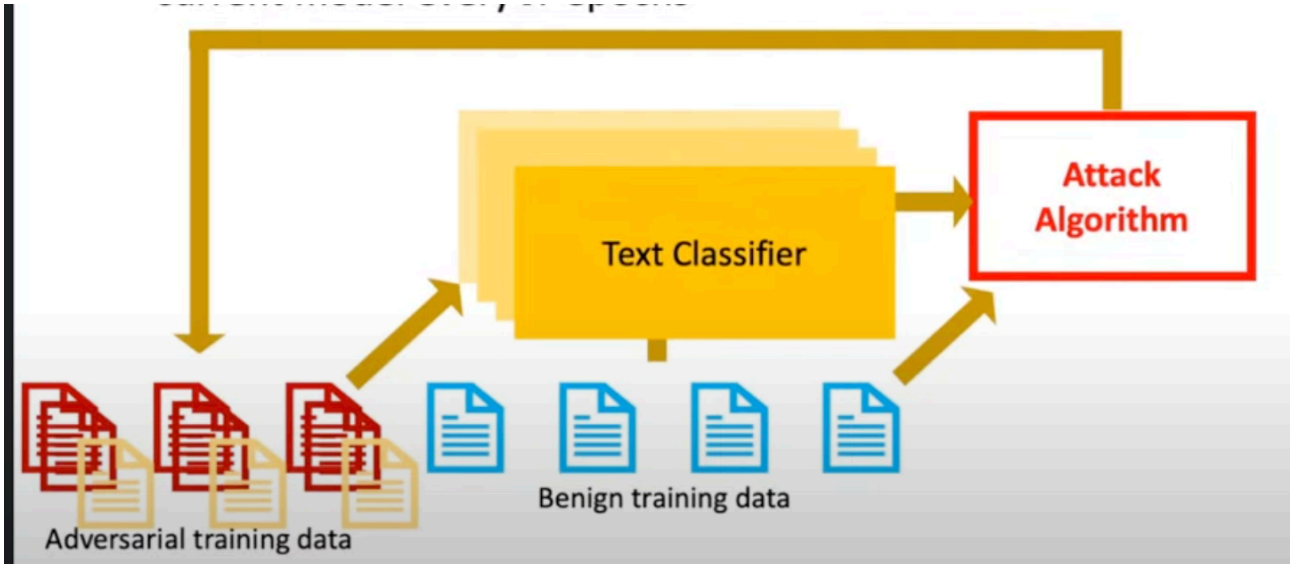


這個例子我將recommend拼成rrecomend，成功攻擊。

那這個例子就是來自助教於上課時提到的，拼錯字就可能會造成判斷錯誤，原因應該也是模型沒見過這個單字，就給出了奇怪的答案，我個人比較好奇的地方是，為何會給一個沒見過的單字如此高的負面權重，於是我測試將highly跟this拼錯，得出來的結果是正面的，並沒有影響判斷，所以我給出的可能解釋是在這個評論中，recommend是確定情緒的重要因素，所以如果出現一個模型沒看過的單字，那給出的結果其實不一定會錯誤，模型可能會亂猜一個答案或者是用很偏頗的資料去判斷(我想Bert如此龐大的training data中應該也有極少數一樣打錯的資料吧)，只是在這個例子中，是猜錯的一方，而我也做了十次相同操作，結果都大同小異，排除了是LIME模型不穩定的因素。

接下來講解如何抵禦這幾個例子：
上面提到的三個例子都能歸屬於evasion attack，就是將原本模型能正確判斷的input用noise去製造出新的input，人類仍會給出相同的答案，但模型卻會判斷錯誤。
Defense的方法有兩種：

一種是補強model，讓他面對攻擊仍能給出對的答案，而這種方法也是最通用的，三個例子都可能被解決。方法很簡單，我們用train data去訓練一個模型，然後我們有一個attack algorithm能產出adversarial samples，而這些example都能成功攻擊模型，那我們再用這些adversarial samples+原本的train data去訓練新的model，然後重複上述的操作直到訓練出我們滿意的model。



上圖是李鴻毅教授課程上的ppt示意圖

這個方法的缺點是attack algorithm的效率太差，導致大家普遍不太喜愛，於是有人提出了另一種方法ASCC-Defense，能夠在訓練過程從原本的data生成類似的句子再拿去訓練，但也很難確定生成出來的一定是類似的句子，或是有包含到所有adversarial samples。

另一種方法就是在丟進model前，能夠先偵測到adversarial attack，並且能夠將其還原，底下給出兩類方法：

Discriminate perturbation(DISP)能針對類似於第二個例子的攻擊，偵測出哪些單字是被修改過的並嘗試還原。

Frequency-Guided Word Substitutions(FGWS)則是能針對一些將高頻字換成低頻字(如第三個例子)的攻擊，偵測句子中不常見的單字並將其轉換成最常見的同義詞再丟到model中。

所遇問題：

不懂如何Defense：
這問題也並沒有太難解決，畢竟每次作業都會遇到滿多不太懂的地方，從一開始完全不會很緊張，到後來學會了不少概念，也懂得看論文跟查資料，還有觀看李宏毅教授的影片，所以這次看完影片之後很快就學會方法並完成作業了，雖說因為一開始完全不會寫python跟不會AI，每次寫作業都查資料查到很痛苦，但確實學會了很多東西，也奠定了我專題方向，相當感謝這門課帶給我的收穫。