自然語言處理概論HW2

0716235徐煜倫

Method 1:

觀察：

為了要從文本中提取SVO進行比對, 首先就要分析給定的字串結構。由於不一定是每一個句型都是SVO結構, 有可能省略賓語或是主語, 所以只要一旦不符合三元組的條件我們都可以去掉。

首先觀察語料結構, 假設這是給定的文字：

I am Alan and I am handsome. I have a nice day.

我們閱讀此段文字就會自己去斷句, 上面這句可以被切割成兩個大句子：

1. I am Alan and I am handsome.
2. I have a nice day .

再來我們可以發現第一句話其實也可以切割成兩句話：

1. I am Alan.
2. I am handsome.

這個精神基本上就是divide and conquer.

方法：

將給定的文本切割成較小較為簡單的片段, 取其SVO就會比較簡單。在找到小短句子之後直接用dep_判斷其為subjects, objects, 還是 pos_ 為 “AUX”,”VERB”。先尋找出句子的動詞（因為觀察parse tree之後的結果, 在spacy中動詞通常作為一句話的根節點），然後從動詞的left children中找出主語，動詞的right children中找出賓語。最後對每段短語句組成(S,V,O)。

判斷：

由於可以取得每個句子的(S,V,O)，但是要和助教給的結果來比較，判斷是否為SVO。這邊使用的方法比較粗糙，因為我取三元組的時候只考慮最短的，並不會包含形容詞

與修飾句，所以只要我的三元組是助教給定的三元組的子集合，我就將label標示為1。

分析：

> F1 score: 0.58（未達標準）

會出現這件事情，是因為我發現助教給的S,V,O中常常參雜其他不屬於此分類的元素。例如V中會出現名詞和介係詞甚至是符號，而我因為判斷相似的方法只有確認是否為子集合，這種方法無法撿測出非法的詞性出現，導致分數低落。

Method 2:

觀察：

有了上個方法的慘烈成績，我開始思考自己的parse策略並不準確的問題。於是我決定利用反面方式來作答。首先先查尋spacy文檔，找出自己的objects & subjects的歸類是否正確，並且更新所有可能的dep標籤：

```
OBJECT_DEPS = {"dobj", "attr", "dative", "oprd"}
SUBJECT_DEPS = {"nsubj", "nsubjpass", "csubj", "agent", "expl"}
```

而後，經過過濾data.csv，我發現除了部份的S,V,O是一些符號或是記號，有超過80%的(S,V,O)都是屬於column “sentence”的子集合。所以我決定利用答案本身去過濾。

方法 ：

方法非常容易，我們只要確認給定的語句分割成短句後，檢查以下幾項事情

1. 一個token的dep標籤屬於OBJECT_DEPS 而且這個token的文字是給定的O欄位的子集合，而且根節點是屬於V欄位的子集合，標記為objExist
2. 一個token的dep標籤屬於SUBJECT_DEPS 而且這個token的文字是給定的S欄位的子集合，而且根節點是屬於V欄位的子集合，標記為subExist

判斷：

只要確認 objExist && subExist 為真，則label標示為1。

分析：

> F1 score: 0.71 (未達標準)

這個方法的準確度有較高的提昇，因為不會被自己的policy漏掉所影響，確保了每個應該出現的S,V,O都可以被正確比對到。不過這方法有一個致命的問題，就是無法偵測錯誤的SVO對應，也就是S,V,O本身隸屬於這段字串，但是不同組別的SVO混在一起依然能被判斷為label=1。

舉例來說：

> 💡 A Spanish official , who had just finished a siesta and seemed not the least bit tense , offered what he believed to be a perfectly reasonable explanation for why the portable facilities were n't in service .

助教給的(S,V,O)是： （ A Spanish official , finished, explanation），這種語句錯亂的例子，label也會因為都滿足條件, 於是標記為1。

Method 3:

特性 & 判斷：

改良第一種方法，使用更好的策略找出所有可能的(S,V,O), 這邊因為不確定noun clause是否作為受詞，所以都使用其代名詞或是wh clause進行代替。這個方法繼承了method 1 的幾個方法：

1. 分割句子 （使用spacy.sent的迭代器分割）
2. 動詞優先的尋找方法

而這個方法額外能做到的：

1. 和助教的語句判斷相似度的時候使用spacy的similarity，並且設置threshold，更有彈性
2. 可以根據找到的subject & object去進行括展，在比對階段將他們的子樹（形容詞和修飾句）包含進來
3. 可解析主詞連接詞子樹，如：Alan and Bob and Mike likes that girl. 的主詞是（Alan, Bob, Mike）
4. 可解析受詞連接詞子樹，如：I love Namin and her sister and her brother. 的受詞是 （Namin, sister, brother）

5. 可分割that代指的對象為兩句話
6. 可替換的WH-clause主詞，如：

> 💡 A Spanish official , who had just finished a siesta and seemed not the least bit tense , offered what he believed to be a perfectly reasonable explanation for why the portable facilities were n't in service .
>
> 這可以回傳 （A Spanish official, finished , siesta) 與 （who, finished , siesta)

方法 (都寫在code的每個函數的註釋裡面了)：

演算法：

```
Find and return SVO triplets.
  1. split the string into sentences
  2. find verbs in a sentence
  3. according to the verb, find joined subjects from its lefthand
  if len(subjects) > 0:
      4. join the right of verb if it is conjunction verb (ex: she hug and kiss me.)
      5. find all objs (要可以替換代名詞, which who ...)
  6. retreive detail dispict of subject and object
  7. append to svo list
```

分析：

> F1 score: 0.75445 （threshold=0.8） （過baseline）

這個方法的確比較好，因為可以利用遞迴解析回原本的句子，使得取得的字串數量較長，也更容易比較相似度。缺點是在給定的（S,V,O）長度較短的時候，因為字數太少了，導致很容易被誤判為相異。舉例來說：

> 給定 V = (walking)

> 而我解析出其中一種V = (has been walking）

這兩個的相似度會被判斷為0.36，但我認為他們其實是一樣的。