Morfeusz i POS-taging

Paweł Rychlikowski

Instytut Informatyki UWr

3 grudnia 2018

Generacja form w Morfeuszu

- Morfeusz działa jako analizator albo generator.
- Ten drugi tryb może być użyteczny, gdy chcemy generować polskie zdania (na przykład przy tłumaczeniu)
- Wyobrażamy sobie, że tłumaczenie jest wieloetapowe i pierwszy etap daje nam wynik typu:

  piękny kobieta kupić różowy sukienka w zielony groszek

  a dokładniej

  (piękny kobieta):sg:nom kupić:sg:f (różowy sukienka)sg:acc w (zielony groszek)pl:acc

Realizacja generacji form

- Można to zrobić na przykład prosząc Morfeusza o podanie wszystkich form dla danego słowa
- a następnie pozostawienie tych, które mają właściwe L,P,R.
- Załóżmy, że chcemy generować frazy takie jak: samotny biały żagiel, ognistych rumaków, śliczną posiadłość, etc.
  Jak zaprojektować interfejs funkcji generującej takie frazy?
- Popatrzmy, jak wygląda implementacja funkcji: `noun_phrase(subst, adjs, number, case)` w Pythonie z wykorzystaniem biblioteki morfeusz.

Hello Word z Morfeuszem

```
import morfeusz2
morf = morfeusz2.Morfeusz()

def get_tags(word):
    res = []
    for beg,end,des in morf.analyse(word):
        orth,base,tag, p1, p2 = des
        res += M._expand_tag(tag)
    return res
```

Metoda _expand_tag jest konieczna, żeby radzić sobie z tagami postaci: a:b.c.d:e, zamieniając je na listę:
'a:b:e', 'a:c:e', 'a:d:e'

Pobranie rodzaju słowa

```
GENDERS = set(['m1','m2','m3','f','n1','n2'])

def gender(noun):
    for tag in get_tags(noun):
        ts = tag.split(':')
        for t in ts:
            if t in GENDERS:
                return t
    return '?'
```

Uwaga

Używane w tagach identyfikatory są unikalne, zatem nie musimy się przejmować kolejnością i **f** zawsze będzie oznaczał rodzaj żeński niezależnie od pozycji, na której wystąpi.

Generacja LPR

```
def genLPR(word, L,P,R):
    candidates = morf.generate(word)
    for w, base, tags, p1, p2 in candidates:
        for tag in morf._expand_tag(tags):
            ts = tag.split(':')
            if L in ts and P in ts and R in ts:
                return w  #UWAGA STOPIEN
    return '?' + word
```

Uwaga

Przymiotniki będą się generowały zawsze w jednym stopniu.

Generator zdań do tej pory

- Wersja z Listy 1: raczej niepoprawne i bezsensowne zlepki słów
- Generator 5-gramowy: odtwarzanie wielkich „połaci” korpusu, czasem „dowcipna” zmiana tematu.

Generator III

- Generujemy zdania postaci:

  Ruda wiewiórka spotkała czarnego kruka.
  Zielony kaktus produkuje smaczne owoce.
  Czarne koty widzą niewesołe perspektywy.
  Piękne książki nie dają pełnej satysfakcji.

- Za poprawność gramatyczną odpowiada Morfeusz (i funkcja nounPhrase) i ...
- lista czasowników tranzytywnych, czyli takich, które akceptują dopełnienie w bierniku (w wersji pozytywnej), albo w dopełniaczu (w wersji negatywnej).

Wersja podstawowa

Chamowate dozowniki poobcierały materialny lepidodendron.
Uczuleniowy instrumentariusz poumieszczał bufonowatego kwakra.
Radialna pielucha opychała niesumienne paragrafowce.
Noworodkowe przemądrzałości pogrążały nadplanową łuskwinę.
Dopełnieniowe żydostwo nie wywinęło kilkunastoprocentowego glifu.
Pięciokrotni nastolatkowie obchodzili bezpowietrzny hemikryptofit.
Pozaresortowe suburbia przemalowały wierzchnie pejzażystki.
Polihistoryczna heurystyka nie podganiała sekwencyjnych trajlusiów.
Endeckie śrutowniki ponadawały fenickie pincety.
Moralne obsypniki zalogowały artezyjskie dobicie.

Trochę bezsensowne te zdania (ale za to poprawne gramatycznie!)

Jak wykorzystać korpus do generacji sensowniejszych zdań?

- Można wykorzystać unigramy i losować częstsze słowa z większym prawdopodobieństwem (i nie losować tych, których nie ma w korpusie)
- Można wykorzystać N-gramy ($N > 1$) i odczytać z korpusu możliwe relacje podmiot-orzeczenie, orzeczenie-dopełnienie, rzeczownik-przydawka, a następnie losować dla wybranego orzeczenia tylko dozwolone podmiot i dopełnienie, a dla nich z kolei dozwolone przymiotniki.
- Jak rozpoznać, że para słów jest realizacją takiej relacji, gdy nie mamy rozbioru?
- Odpowiedź: można się nie przejmować globalnym rozbiorem tylko sprawdzać spełnienie lokalnych warunków gramatycznych.

Generator+

- Na kolejnej liście będzie zadanie z prośbą o generowanie lepszych zdań dzięki realizacji wcześniej wymienionych postulatów.
- Oczywiście można łączyć z Panem Tadeuszem!

Part of Speech Tagging. Przypomnienie zadania

Dane wejściowe

Na wejściu mamy:

1. Tekst, podzielony na tokeny
2. Informacje o tym, jakie są możliwe znaczniki dla każdego tokenu (słownik morfosyntaktyczny).

Wynik

Wyjściem jest przypisanie dla każdego tokenu optymalnego znacznika (tagu) w danym kontekście.

Czyli sekwencja słów zostaje wzbogacona o sekwencję tagów optymalnie opisującą ten ciąg.

Przykład otagowanego korpusu (KPWR)

Zatrzasnął/praet:sg:m1:perf:ter drzwi/subst:pl:acc:n od/prep:gen
mieszkania/subst:sg:gen:n ,/interp dwa/num:pl:acc:m3:congr
razy/subst:pl:acc:m3 przekręcił/praet:sg:m1:perf:ter
klucz/subst:sg:acc:m3 ,/interp nacisnął/praet:sg:m1:perf:ter
klamkę/subst:sg:acc:f ,/interp by/conj sprawdzić/inf:perf ,/interp
czy/qub dobrze/adv:pos zamknięte/ppas:pl:nom:n ,/interp
zbiegł/praet:sg:m1:perf:ter po/prep:loc schodach/subst:pl:loc:n ,/interp
minął/praet:sg:m1:perf:ter furtkę/subst:sg:acc:f ,/interp także/qub
ją/ppron3:sg:acc:f:ter:akc:npraep zamknął/praet:sg:m1:perf:ter ,/interp
i/conj znalazł/praet:sg:m1:perf:ter się/qub na/prep:loc
wąskiej/adj:sg:loc:f:pos uliczce/subst:sg:loc:f między/prep:inst
ogródkami/subst:pl:inst:m3 ,/interp gdzie/adv
drzemały/praet:pl:m3:imperf:ter w/prep:loc majowym/adj:sg:loc:n:pos
słońcu/subst:sg:loc:n trójkątne/adj:pl:nom:m3:pos
ciemnozielone/adj:pl:nom:m3:pos świerki/subst:pl:nom:m3 ,/interp
jakich/adj:pl:gen:m3:pos nie/qub było/praet:sg:n:imperf:ter w/prep:loc
pobliżu/subst:sg:loc:n jego/ppron3:sg:gen:m1:ter:akc:npraep
domu/subst:sg:gen:m3 ./interp

Przykład otagowanego korpusu (KPWR)

Zatrzasnął/praet:sg:**m1**:perf:ter drzwi/subst:pl:**acc**:n od/prep:gen
mieszkania/subst:**sg:gen**:n ,/interp dwa/num:pl:**acc:m3**:congr
razy/subst:pl:**acc**:m3 przekręcił/praet:sg:**m1**:perf:ter
klucz/subst:sg:**acc**:m3 ,/interp nacisnął/praet:sg:**m1**:perf:ter
klamkę/subst:sg:acc:f ,/interp by/conj sprawdzić/inf:perf ,/interp
czy/qub dobrze/adv:pos zamknięte/ppas:**pl:nom:n** ,/interp
zbiegł/praet:sg:**m1**:perf:ter po/prep:**loc** schodach/subst:pl:loc:n ,/interp
minął/praet:sg:**m1**:perf:ter furtkę/subst:sg:acc:f ,/interp także/qub
ją/ppron3:sg:acc:f:ter:akc:npraep zamknął/praet:sg:**m1**:perf:ter ,/interp
i/conj znalazł/praet:sg:**m1**:perf:ter się/qub na/prep:**loc**
wąskiej/adj:sg:**loc**:f:pos uliczce/subst:sg:loc:f między/prep:inst
ogródkami/subst:pl:inst:m3 ,/interp gdzie/adv
drzemały/praet:pl:**m3**:imperf:ter w/prep:**loc** majowym/adj:**sg:loc:n**:pos
słońcu/subst:sg:loc:n trójkątne/adj:**pl:nom:m3:pos**
ciemnozielone/adj:**pl:nom:m3**:pos świerki/subst:pl:**nom**:m3 ,/interp
jakich/adj:pl:**gen:m3**:pos nie/**qub** było/praet:sg:n:imperf:ter w/prep:**loc**
pobliżu/subst:sg:loc:n jego/ppron3:sg:gen:**m1**:ter:akc:npraep
domu/subst:sg:gen:m3 ./interp

Jeszcze o otagowanym korpusie

- Na poprzednim slajdzie **pogrubione** były miejsca, w których tager musi podjąć decyzję.
- Zwróćmy uwagę, że otagowany korpus daje (wraz z generowaniem form) możliwość tworzenia zdań zgodnych ze schematem.

Zbiory tagów jako SuperTagi

- Tagi dla przymiotnika wyglądają (w uproszczeniu) tak:

  adj:pl:nom:m2
  adj:pl:nom:m3
  adj:sg:gen:f
  (...)

- Czyli mają postać: adj:L:P:R
- Mamy 2 liczby, 5 rodzajów, 6 przypadków (bo wołacz jest zawsze równy mianownikowi, mój piękny chłopcze!)
- Czyli jest 60 różnych tagów i 2^{60} potencjalnych zbiorów tagów. A ile jest ich naprawdę?
- piękny, piękna, piękne, piękni, pięknego, pięknej, pięknemu, piękną, pięknym, pięknych, pięknej (11)

Zbiory tagów jako SuperTagi (2)

- Identyfikatory T237 z pliku supertags.txt oznaczają zbiory tagów.
- Supertagi są dużo bardziej jednoznaczne (a w wielu sytuacjach wystarczające).
- każdej z 11 form przymiotnika opowiada dokładnie 1 supertag!

Tagowanie a rekonstrukcja samogłoskowa. Przypomnienie

- W rekonstrukcji samogłoskowej szukaliśmy

$$W' = \text{argmax}_W P(W)$$

  przeglądając wszystkie dozwolone sekwencje słów W.
- $P(W')$ obliczaliśmy korzystając z bigramowego modelu języka, określającego następstwa słów.
- Tym razem mamy do czynienia z sekwencją tagów (której szukamy) i sekwencją słów (która jest dana).
- Podobnie jak poprzednio, możemy mówić o dozwolonej sekwencji tagów (bo mamy ciąg słów oraz słownik).

Twierdzenie Bayesa i tagowanie

- Naszym celem jest obliczenie optymalnej sekwencji tagów T':

$$T' = \text{argmax}_T P(T|W)$$

  dla danego ciągu słów W.
- Z twierdzenia Bayesa mamy:

$$P(T|W) = \frac{P(W|T)P(T)}{P(W)}$$

- Widzimy, że:
  - Pojawił się czynnik $P(T)$, który nawiązuje do modelowania języka traktowanego jako sekwencja tagów.
  - Mamy czynnik łączący słowa z tagami, czyli $P(W|T)$
  - Czynnik $P(W)$ jest nieistotny (bo?)

Tagowanie (cd)

Celem naszym jest policzenie:

$$T' = \text{argmax}_T P(T|W) = \text{argmax}_T \frac{P(W|T)P(T)}{P(W)}$$

po uwzględnieniu stałego $P(W)$, mamy:

$$T' = \text{argmax}_T P(W|T)P(T)$$

Jak szacować te wielkości?

- $P(T)$ możemy szacować tak jak robiliśmy to dla bigramowego modelu języka (z tym, że teraz modelujemy sekwencje tagów, a nie słów, ale otagowany korpus daje do tego dane)
- $P(W|T)$ szacujemy jako:

$$\prod_i p(w_i|t_i)$$

  zakładając (zapewne niesłusznie) niezależność rozkładów.
- $p(w|t) = \frac{C(w,t)}{C(t)}$, czyli sprawdzamy, jaki procent tagów t w korpusie to jest słowo w.

Obliczanie optymalnej sekwencji tagów

- Mamy do policzenia następującą sekwencję tagów:

$$\text{argmax}_T P(W|T)P(T)$$

czyli

$$\text{argmax}_{(t_1,\ldots,t_n)}(\prod_{i=1,\ldots,n} p(w_i|t_i) \times \prod_{i=1,\ldots,n} p(t_i|t_{i-1}))$$

- Możemy przyjąć, że t_0 to specjalnie dodany sztuczny tag [START].
- Oczywiście możemy (i powinniśmy) przejść na logarytmy i sumowanie.

Obliczanie optymalnej sekwencji tagów (cd)

- To jest bardzo podobne, jak w przypadku zadania samogłoskowego (z drobnymi jedynie różnicami):
  - Tam mieliśmy warianty słów (generowane ze słownika), tu warianty tagów (ze słownika morfosyntaktycznego).
  - W obliczeniach zmiennej `cost` (to w zasadzie powinien być profit, ale że ujemny, to...) musimy dodatkowo uwzględnić składową unigramową ($\log p(w_i|t_i)$)

Ten algorytm nazywa się Algorytmem Viterbiego i jest bardzo często używany w różnych zadaniach NLP.

Tager trigramowy

- Czasami rozważa się tager trigramowy. Wówczas zmieniony sposób obliczania $P(t)$ na korzystający z trigramów:

$$P(T) = \prod_{i=1,\ldots,n} p(t_i|t_{i-2}t_{i-1})$$

- Wówczas patrzymy nie na ostatni tag, lecz na parę ostatni i przedostatni i dla każdej takiej (możliwej) pary musimy pamiętać i dla każdego miejsca w ciągu musimy pamiętać wartość optymalnej ścieżki kończącej się w tym miejscu tą parą.
- Możemy dołożyć dwa sztuczne tagi na początku: [START0] oraz [START1].

Na ile intuicyjny jest tager bigramowy?

Jak, nie wiedząc o twierdzeniu Bayesa, zrobilibyśmy tager:

1. Premiowalibyśmy sytuacje, w której w sekwencji pojawiają się tagi często występujące obok siebie w korpusie
2. Pewnie zauważylibyśmy, że pewne słowa (musi, potem, miał, etc) mają zdecydowanie częściej jeden tag niż inne w słowniku. Ten częsty tag premiowalibyśmy.

Czy otrzymalibyśmy te same wzory?

Na ile intuicyjny jest tager bigramowy?

- $f(p(t_{i+1}|t_i))$ albo $f(p(t_i t_{i+1}))$
- $f(p(t_i|w_i))$ albo $f(p(t_i w_i))$ albo $f(p(w_i|t_i))$

dla pewnej rosnącej funkcji f.
Co ciekawe, dla połączenia tag-słowo, pierwsze tagery wykorzystywały właśnie $p(t_i|w_i)$, czyli de-facto $p(t_i, w_i)$

A co jeżeli nie mamy otagowanego korpusu?

- Jakieś dane są potrzebne, przyjmijmy zatem, że mamy po prostu duży zbiór tekstów oraz słownik morfosyntaktyczny.
- Będziemy powtarzać iteracyjnie 2 fazy:
  1. Znajdujemy optymalne tagowanie dla każdego zdania w korpusie (Faza M)
  2. Obliczamy $p(t|t')$ oraz $p(w|t)$ na podstawie tak otrzymanego korpusu (Faza E)
- Mamy nadzieję, że korpus będzie coraz lepiej otagowany, co sprawi, że coraz lepiej oszacujemy prawdopodobieństwa, co sprawi, że lepiej otagujemy korpus, co sprawi, że...
- Taki algorytm należy do klasy algorytmów typu EM (Expectation-Maximalisation)

Punkt startowy

- Czasem używa się czegoś nazywanego flat start, czyli zakłada, że wszystkie prawdopodobieństwa są równe, ewentualnie z jakimiś drobnymi zaburzeniami.
- Ale my możemy zrobić lepiej: (jak?)

Lepszy punkt startowy

- Chcemy zliczać pary tagów, które wystąpiły.
- Popatrzmy na zdanie:

  Potem spraw mam wiele.

- Dla bigramu **Potem spraw** nie wiemy, którą z poniższych par on będzie:
  subst-imp, subst-subst, adv-subst, adv-imp
- Licznik dla każdej z par zwiększamy o 0.25

Inne pomysły na tagowanie

- Tager unigramowy (dla języka angielskiego ma: **90%**)
- Reguły zdefiniowane przez lingwistów
- Uczenie się reguł (tager transformacyjny)
- Inne tagery statystyczne (MaxEnt, MaxEnt HMM, Conditianal Random Fields, ...)
- Sieci neuronowe i modele seq2seq (używane między innymi w tłumaczeniu)
- I wiele innych (dlaczego?)

Przepis na tager

1. Przyjmij, że Twój tager będzie działał od lewej do prawej.
2. Wybierz swój ulubiony mechanizm uczenia automatycznego (drzewa decyzyjne, SVM, Naive Bayes, sieci neuronowe, ...). Każde wystąpienie słowa w tekście to będzie zadanie klasyfikacji, w którym na podstawie pewnych cech mechanizm podejmie decyzję odnośnie taga.
3. Cechy wyznaczysz z otoczenia klasyfikowanego słowa i już przypisanych tagów na lewo od niego.
4. Ucz mechanizm w oparciu o otagowany korpus.
5. Korzystaj do woli z nauczonego tagera, pilnując, by zawsze wybierać najlepiej oceniany tag **z tych, które dopuszcza słownik morfosyntaktyczny**.

Przykładowe cechy

Rozważamy zdanie:

Stół miał szlachetną sylwetkę.

Przykładowe cechy użyteczne w tym przypadku:

- Słowo jest pisane wersalikiem (może rzeczownik?)
- Słowo jest pierwsze w zdaniu (może podmiot?)
- W bliskiej okolicy słowa jest rzeczownik (potencjalnie) w mianowniku i rzeczownik w bierniku. (może orzeczenie?)

Tager regułowy

- W poprzednim slajdzie zakładaliśmy „radosną twórczość regułową”
- Można reguły tworzyć staranniej, by opisywały możliwie dokładnie jakąś sytuację w zdaniu i określały, jaki wówczas powinien być użyty tag.
- Jeden z lepszych tagerów dla języka angielskiego to: **EngCG**

Tager EngCG

- Tager zawiera słownik morfosyntaktyczny, pierwszą fazą operacji jest odczytanie zbiorów tagów właściwych każdemu słowu.
- Tager zawiera 3700 reguł, które umożliwiają usuwanie pewnych tagów niepasujących do kontekstu.
- Reguły mają postać:
  if warunek **then** akcja1 **else** akcja2
- Warunki są koniunkcją warunków elementarnych i odnoszą się do konkretnego słowa w tekście.
- Warunki elementarne składają się z:
  1. Specyfikacji miejsca (+1, -1, +2, -2), podającej pozycję względem słowa bieżącego
  2. Treści, która mówi o należeniu słowa do zbioru albo o posiadaniu przez słowo w bieżącym zbiorze tagów określonych tagów.
- Akcje mówią o usunięciu jakiegoś tagu z dopuszczalnych w danym kontekście.

Tworzenie reguły

Zadanie

Stworzyć regułę, która umożliwi poprawne tagowanie takich słów jak **spraw**, czyli rozkaźników i rzeczowników w dopełniaczu liczby mnogiej jednocześnie:

> *Spraw się dobrze!*
> *Ministerstwo Spraw Zagranicznych oświadcza, że ...*

Jakie to są słowa?
Spraw, zestaw, zgraj, zastaw, napraw, ...

Przykładowe reguły

Z1: **Zgraj** *to na płytę!*
Z2: *Janku,* **napraw** *proszę ten opiekacz.*
Z3: *Mam dość tych ciągłych* **napraw***!*

- Jeżeli początek zdania (czyli na pozycji -1 koniec poprzedniego zdania) to wówczas rozkaźnik. (Z1)
- Jeżeli poprzednie słowo to przecinek, a jeszcze poprzedniejsze rzeczownik liczby pojedynczej w wołaczu, a po nim następuje słowo proszę, to również rozkaźnik. (Z2)
- Jeżeli na pozycji -2 jest słowo „tych", a na pozycji -1 przymiotnik w dopełniaczu liczby mnogiej, to rzeczownik (Z3).

EngCG – podsumowanie

- EngCG czasami zostawia więcej niż 1 tag – wówczas można albo pozostawić niepewność, albo wziąć najczęstszy tag danego słowa (z korpusu)
- Po zastosowaniu reguł 93-97% zostaje w pełni zdezambiguowanych
- a 99.7 zawiera właściwy tag.

Ale...

Tworzenie takich reguł jest bardzo kosztowne!

Uczenie się reguł

- Naturalne jest pytanie, czy możemy się uczyć reguł tagowania (z otagowanego korpusu)
- Odpowiedzią jest Brill Tager, czyli tager transformacyjny.
- Brill tager zaczyna działanie od tagowania bazowego, a następnie próbuje je poprawić (zapamiętując, jak to zrobił)

Uwaga

EngCG odejmował tagi ze zbioru, Brill tager pamięta zawsze najlepszy tag dla każdego wystąpienia słowa.

Tager transformacyjny

- Tagowanie bazowe to przypisanie najczęstszego tagu dla każdego słowa.
- W pojedynczym kroku procedury uczącej sprawdzana jest jedna reguła (zakładamy, że mamy sposób iterowania reguł).
- Wśród reguł poprawiających tagowanie wybieramy tę, która daje największą poprawę i ją zapisujemy na liście reguł.

Tager transformacyjny (2)

- Umawiamy się na postać reguły, na przykład takie

```
Tag[0] = t1 & Tag[p1] = t2 ==> ChangeTagFor0(t3)
Pos[0] = pos1 & Tag[p1] = t1
                  & Tag[p2] = t2 ==> ...
Word[p1] == w1 & Word[p2] == w2 ==> ...
        ...
```

  gdzie t1, t2 to tagi, p1,p2 to pozycje względem słowa tagowanego ({ -2,-1,1,2}

- Słowa należą do niewielkiego zbioru słów specjalnych (żeby liczba reguł nie wzrosła zanadto)
- Reguła jest stosowalna, jeżeli zmiana tagu jest aprobowana przez słownik

Tager transformacyjny (3)

- Tagerem jest lista reguł zapisana w etapie uczenia.
- Reguły te będą potem wykonywane na nieznanym tekście w zapisanej kolejności (oczywiście zaczynamy od wyboru najczęstszych tagów)
- Znając reguły możemy próbować jakoś je skompresować, przy zachowaniu semantyki.
- Można łączyć z innymi tagerami? (jak?)
- na przykład rozpoczynać nie od wyniku tagera unigramowego, lecz bigramowego.