Przetwarzanie języka naturalnego
Pracownia 4

Wszystkie zadania są ważne do końca semestru, zadania oznaczone literką **E** można oddawać również podczas dodatkowego terminu przed egzaminem.

Zadania w większości są z gwiazdką (która nie jest zaznaczana, niegwiazdkowych zadań jest za 10 punktów). Będzie jeszcze druga część listy (już cała z gwiazdkami), zadania na niej są podane jako zapowiedzi). Na tej liście znajdą się również punkty za zadania z rekonstrukcją (permutacje, ogonki i małe/wielkie litery) oraz nowe dane do zadania ze zgadywaniem autorstwa tekstu (wraz z informacją o punktacji).

Osoba, która rozwiąże zadania z literką **E** za co najmniej 10 punktów może uzyskać zwolnienie z egzaminu i przepisanie oceny z ćwiczeń, jeżeli ta ocena jest co najmniej 4. Jeśli chodzi o zaliczanie ćwiczeń, to zadania z E działają jak normalne zadania.

Zadanie 1. (**4p**) Zmodyfikuj program `np.pl` tak, by był w stanie rozpoznać 8000 fraz nominalnych (w programie tym jest zaszyty warunek, że fraza nominalna ma mniej niż K wyrazów, jego (w tym zadaniu) nie powinieneś zmieniać.

Zadanie 2. (**4p**) Napisz program wykorzystujący bibliotekę NLTK, by był w stanie rozpoznać 8000 fraz nominalnych. To, jak tego typu gramatyki są wykorzystane w NLTK można przeczytać w rodziale Building Feature Based Grammar, (`http://www.nltk.org/book/ch09.html`).

Zadanie 3. (**1-6, ⋆, Ep**) W zadaniu tym możesz korzystać z dowolnego języka, w którym jesteś w stanie napisać parser taki, jak w poprzednich zadaniach (w szczególności możesz korzystać rozwiązań poprzednich zadań. Możesz również korzystać z informacji „wyciągniętych" ze słownika walencyjnego (Walentego), które znajdziesz na stronie wykładu. W zadaniu chodzi o to, żeby poprawić jakość parsera. Mały punkcik otrzymujemy za sparsowanie poprawnej frazy. Dodatkowo na stronie wykładu jest lista fraz, które nie są frazami **np**, zaakceptowanie którejś z nich oznacza 100 punkcików kary[1].

Punkciki na punkty przekładają się w następujący sposób:

```
8300   1p
9000   2p
9500   3p
10000  4p
10500  5p
11000  6p
```

Przekroczenie liczby 11000 będzie dodatkowo wynagradzane, wg wzoru $\sum_{i=1}^{K} 0.2 \times 0.9^{i-1}$, gdzie K jest liczbą pełnych setek przekraczających 11000. Uwaga: reguły gramatyki powinny opisywać albo ogólne prawidłowości, albo wyjątki, nie jest dozwolone opisywanie jako wyjątkowych tych zjawisk, które da się opisać w sposób bardziej ogólny[2].

Zadanie 4. W tym zadaniu będziemy rozważać probabilistyczne gramatyki bezkontekstowe (PCFG) utworzone ze Składnicy. Została przygotowana wersja składnicy w formacie akceptowanym przez NLTK, zawierającym informacje o parametrze *head* dla każdej frazy. Programik `skladnica_demo.py` może stanowić pomoc w przetwarzaniu takich drzew. Analizując ten bank drzew utwórz gamatykę PCFG, którą następnie wykorzystamy do generowania zdań. Rozważamy następujące warianty:

a) Wariant w pełni zleksykalizowany: symbolem nieterminalnym jest symbol nieterminalny z wszystkimi parametrami oraz z parametrem head, przykładowo: `ff:poj:2:[np[bier]]|wypijesz`. Zauważ, że powtarzanie pewnych produkcji tej gramatyki powoduje tworzenie nienaturalnych zdań, takich jak:

[1] Lista fraz negatywnych jest utworzona automatycznie i potencjalnie mogą znajdywać się na niej błędy – jeżeli uważasz, że coś na tej liście jest poprawną i naturalną frazą nominalną, zgłoś to na forum dla tej pracowni

[2] Chodzi o to, że nie wolno stworzyć reguły, która mówi, że frazą nominalną jest wszystko to, co znajduje się w pliku phrases.pl, ewentualnie to, co ma te same tagi jak frazy w phrases.pl

który w szczecinie załatwia ważne problemy , istotne dla sądu dla wsi dla użytkowników
człowiek staje się wyleniałym wyleniałym wyleniałym wyleniałym wyleniałym tygrysem
proponuję przyjąć decyzję o wyrwanej wyrwanej kartce z kolei z historii

Postaraj się jakoś temu zapobiec.

b) **(3p)** W tym wariancie powinieneś zrezygnować z leksykalizacji (czyli symbolem nieterminalnym będzie, przykładowo, `ff:poj:2:[np[bier]]`. Dodatkowym parametrem powinna być liczba N, mówiąca jak długie (w przybliżeniu) powinno być zdanie. Dodatkowo powinieneś premiować zdania „rozłożyste", czyli takie, w których drzewo rozbioru jest niezbyt głębokie. Oczywiście najprościej zrealizować to losując wielokrotnie i wybierając takie zdanie, które najlepiej odpowiada kryteriom.

c) **(3p)** Sporządź listę typów używanych przez Składnicę (czyli przypisanie słowom *napisów w nawiasach kwadratowych*, takich jak `[np[bier]]` dla słowa *wypijesz*). Zmodyfikuj losowanie z poprzednich dwóch punktów w ten sposób, by w miejscu słowa o określonym typie mogło pojawić się inne słowo o tym samym typie. Dodatkowo w ostatecznym zdaniu słowa, dla których użyto typu pustego (`[]`) powinny być oznaczone gwiazdką.

d) **(3p, E)** Wzbogać losowanie zdań o zamianę słów z gwiazdką z poprzedniego podpunktu na słowa ze słownika o takim samym tagu (użyj pliku supertags.txt, w nowszej wersji). W losowaniu powinieneś premiować słowa „pasujące" (pasowanie możesz zdefiniować dowolnie, za pomocą bigramów, PMI, zanurzeń słów, etc).

Zadanie 5. (7p, E) W zadaniu będziemy ponownie losować wersy Pana Tadeusza, ale tym razem nie korzystając z oryginału, lecz tylko z korpusu PolEvala oraz z zanurzeń wektorowych. Dla ułatwienia przygotowany został zbiór *poprawnych rytmicznie*[3] zdań z PolEvala (zawierających tylko słowa z pliku supertags). Większość z nich się nie rymuje (ale dla ułatwienie w pliku z tymi zdaniami zawarte są tylko takie, które da się zrymować, z zachowaniem liczby sylab i tagów gramatycznych ostatnich słów).

a) Napisz program losujący dwuwers i modyfikujący ostatnie wyrazy (być może nie oba) w wersach w ten sposób, by się rymowały (z zachowaniem tagu i liczby sylab). W wyborze powinieneś premiować sytuację, w których wyrazy po zrymowaniu są podobne do wyrazów oryginalnych (czyli wektory ich form bazowych mają możliwie duży iloczyn skalarny). Nie dla każdego dwuwersu to da się zrobić, powinieneś to uwzględniać przy wyborze dwuwersu. Przykładowy wynik działaniu programu (potencjalnie użyteczny do testów):

ORYGINAŁ: po zjednoczeniu niemiec dotychczas strzeżony [*] obszar został otwarty i przebudowany .
POEZJA: po zjednoczeniu niemiec dotychczas strzeżony [*] obszar został otwarty i podpiwniczony .

b) Dodaj do powyższego programu możliwość zamiany wybranych słów na inne. Zamieniać powinieneś tylko czasowniki, rzeczowniki, przysłówki, imiesłowy i przymiotniki. Oczywiście w zamianie zachowujemy tag i staramy się zachować podobieństwo do oryginału. Nie wolno nam też zepsuć rymu. Przykładowe działanie:

ORYGINAŁ: seria dziecięcych skarpet antypoślizgowych [*] z motywami mieszkańców obszarów polarnych .
POEZJA: seria dziecięcych skarpet antypoślizgowych [*] z motywami mieszkańców obszarów szelfowych .
ZMODYFIKOWANA: seria przedszkolnych skarpet antypoślizgowych [*] z pejzażami parafian terenów magmowych .

[3] Mają one 26 sylab z przerwami po 7, 13 i 20 sylabie, na końcu każdego wersu i przed średniówkami nie ma słów jednosylabowych

Zadanie 6. (10p, E) Stwórz możliwie prostą, a jednocześnie nietrywialną gramatykę generującą zdania w języku polskim (zdania powinny być poprawne gramatycznie, możliwie naturalne, możesz korzystać z typów słów ze Składnicy, na SKOSie pojawi się też lista czasowników tranzytywnych[4]). Możesz stosować pomysły i kod z poprzednich zadań na tej liście. Do generowania słów możesz wykorzystać Morfeusza (zob. wykład 7).

Do nadawania sensu generowanym zdaniom wykorzystaj następujący mechanizm:

1. Wejściem do generatora powina być liczba N (liczba wyrazów w zdaniu, traktowana jako „wskazówka odnośnie wielkości zdania") oraz zbiór słów, stanowiących: *osnowę historii*. Przykładowe osnowy to: malina-koszyk-zazdrość-morderstwo, programowanie-błąd-zmienna-deklaracja, lotniskowiec-łódź-podwodny-tonąć-atak-torpeda-ocean.

2. Pierwszym etapem generowania zdania jest generowanie zdania poprawnego gramatycznie (bez związku z osnową)

3. Następnie dla każdego wyrazu, który jest rzeczownikiem, przymiotnikiem, imiesłowem, czasownikiem, przysłówkiem generujemy kilkaset jego wariantów i wybieramy taki, który pasuje najlepiej do **jakiegoś** słowa z osnowy. Wykorzystujemy tu zarówno zanurzenia słów i/lub form bazowych.

4. Proces powtarzamy wiele razy, wybierając zdanie, które wydaje się najlepsze (na przykład wysokie jest wzajemne PMI poszczególnych par słów oraz czy wszystkie słowa z osnowy zmieściły się w zdaniu)

Planem minimum jest zapewnienie, żeby było łatwo zgadnąć, którą z trzech przykładowych osnów realizuje wygenerowane zdanie. Możesz to przetestować na sobie, generując na przykład 30 zdań (dla każdego losując wcześniej jedną z trzech osnów, zapisując losowanie do pliku i sprawdzając, czy jesteś w stanie odtworzyć ten plik patrząc tylko na wyniki losowań)

Zadanie 7. (**7+X, Ep**) Wykorzystaj wersję zależnościową Składnicy ze strony Universal Deppendencies, aby generować zdania. Twój mechanizm powinien:

a) Umożliwiać podstawianie za liście słów o podobnych gramatycznie.

b) Uwzględniać typy wyrazów (czyli to, jakie tagi mają ich dzieci, kolejność dzieci i ich położenie względem rodzica)

c) Generować zróżnicowane zdania

d) Generować poprawne gramatycznie zdania.

e) Uwzględniać fakt, że niektóre typy są „uniwersalne", czyli możliwe jest ich stosowanie do dowolnych wyrazów o określonym tagu.

Szczegóły pozostawione są do dopracowania przez Studenta. Wartość X jest nieujemna, zależy od złożoności rozwiązania. Domyślnie jest równa 0.

Zadanie 8. (**10, Ep**) Generuj poezję za pomocą gramatyk PCFG. Poezja może wyglądać jak Pan Tadeusz, albo jak piosenka Tanie Dranie. Konieczne jest zadbanie o poprawność rytmiczną i rymy oraz o poprawność gramatyczną. Należy również w jakiś sposób premiować zdania sensowne treściowo.

Zadanie 9. (**10, Ep**) Zapowiedź: generuj poezję za pomocą gramatyk zależnościowych. Poezja może wyglądać jak Pan Tadeusz, albo jak piosenka Tanie Dranie. Konieczne jest zadbanie o poprawność rytmiczną i rymy oraz o poprawność gramatyczną. Należy również w jakiś sposób premiować zdania sensowne treściowo.

[4]Czyli takich, które „obsługują" dopełnienie w bierniku i dopełniaczu w wersji zanegowanej (czyła książki, nie czytał gazet)

Zadanie 10. (**15+, Ep**) Zapowiedź: generuj poezję korzystając z rozszerzonego banku wzorców zdań z PolEvala (utworzenie jego jest częścią zadania). Rozszerzenie polega na tym, że zdanie wejściowe nie musi być poprawne rytmicznie, ale powinno być możliwe jego „urytmicznienie" (to znaczy zamiana pewnych słów na inne, o innej liczbie sylab, przy częściowym zachowaniu sensu zdania).

Dodatkową daną dla generatora powinna być *osnowa* wiersza(zobacz zadanie 6). Uwaga: poziom trudności tego zadania wydaje mi się oscylować w okolicach pracy inżynierskiej, z całkiem sporym wow-faktorem.