

Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Radomir Mastalerz

Nr albumu: 236104

Tager maksimum entropii dla języka polskiego

**Praca magisterska
na kierunku INFORMATYKA**

Praca wykonana pod kierunkiem
dr. hab. Adama Przepiórkowskiego
Wydział Matematyki, Informatyki i Mechaniki
Uniwersytet Warszawski

Luty 2011

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data

Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data

Podpis autora (autorów) pracy

Streszczenie

Niniejsza praca magisterska zawiera opis zastosowania metody maksimum entropii do tagowania tekstów z Narodowego Korpusu Języka Polskiego. Metoda została wcześniej z sukcesem zastosowana do tagowania słów z języka angielskiego i czeskiego, natomiast nie była nigdy stosowana dla języka polskiego. Tager Ratnaparkhiego dla języka angielskiego osiągnął skuteczność 96,6%, a tager Hajiča dla języka czeskiego 93,78%. Opisany w tej pracy tager dla języka polskiego osiągnął skuteczność 93,1% dla tagowania pełnych tagów. Jest to jeden z najlepszych wyników uzyskanych dla języka polskiego. Dla porównania — najnowszy tager Szymona Acedańskiego, opisany w pracy [Acedański 2010], osiągnął skuteczność 92,82%.

Słowa kluczowe

MaxEnt, tager maksimum entropii, model log-normalny, NKJP, Narodowy Korpus Języka Polskiego

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna

I. Computing Methodologies
I.2 Artificial Intelligence
I.2.7 Natural Language Processing

Tytuł pracy w języku angielskim

Maximum entropy tagger for Polish

Spis treści

Przedmowa	5
1. Wstęp	7
1.1. Zadanie tagowania	7
1.2. Korpus	9
1.3. Tagset	9
1.4. Analizator morfologiczny Morfeusz	10
1.5. Krótko o modelu maksimum entropii	11
2. Model maksimum entropii od strony teoretycznej	13
2.1. Entropia	13
2.2. Intuicje dotyczące zasady maksimum entropii	14
2.3. Model tagera	15
2.4. Cechy kontekstowe	15
2.5. Zastosowanie zasady maksimum entropii do wyboru tagera	17
2.6. Zasada maksimum entropii a zasada największej wiarygodności	20
3. Wybór cech kontekstowych	23
3.1. Dwuetapowy wybór tagu dla słowa	23
3.2. Klasy niejednoznaczności	24
3.3. Przykład użycia modelu do wyboru tagu dla słowa	25
3.4. Użyte w modelu schematy cech kontekstowych	27
3.4.1. Części mowy	28
3.4.2. Kategorie morfologiczne	28
4. Ewaluacja tagera	31
4.1. Metodologia i dane treningowe	32
4.2. Skuteczność modelu maksimum entropii	32
4.3. Tager MaxEnt dla języka czeskiego	34
4.4. Porównanie skuteczności tagerów dla języka polskiego	35
4.5. Analiza błędów	36
4.6. Pomysły na usprawnienie tagera	38
5. Implementacja	41
Zakończenie	43

A. Metoda mnożników Lagrange’a	45
A.1. Intuicje geometryczne	45
A.2. Algorytm	47
B. Dokumentacja użytkowa programu	49
B.1. Instalacja	49
B.2. Opis poszczególnych programów i zawartości plików	50
C. Zawartość załączonej płyty CD	53
Bibliografia	55

Przedmowa

Tagowanie to jedno z ważniejszych zadań przetwarzania języka naturalnego. Zadaniem tagera (maszyny służącej do tagowania) jest przypisanie każdemu wyrazowi w tekście wejściowym właściwych znaczników morfosyntaktycznych. Znaczniki morfosyntaktyczne (tzw. tagi) zawierają informacje określające część mowy tagowanego słowa (np. czasownik, rzeczownik, przymiotnik) oraz wartości kategorii morfologicznych, takich jak: przypadek, liczba, rodzaj, osoba. Tager wybiera dla słowa część mowy oraz odpowiednie wartości kategorii morfologicznych, bazując na kontekście, w jakim znalazło się tagowane słowo.

Wyobraźmy sobie stworzenie idealnego modelu do tagowania. Jak on mógłby wyglądać? Przyporządkowując tag do słowa w , chcielibyśmy wykorzystać pełny kontekst słowa w , tzn. wszystkie dostępne na moment tagowania informacje. Ciężko zdefiniować pojęcie *wszystkich dostępnych na moment tagowania informacji*. Do takich informacji na pewno możemy zaliczyć tagowane słowo w , słowa w otoczeniu w , tagi przypisane przez model poprzednim słowom (o ile tagujemy od lewej do prawej), może również słownik najbardziej prawdopodobnych sekwencji tagów, słownik najczęściej występujących w tekście związków wyrazowych, metainformacje o tekście. Ciężko postawić tutaj jakiekolwiek granice. Tager, napotykając nowe słowo, do wyboru odpowiedniego tagu może użyć również charakterystycznych cech wynikających z kontekstu, np. końcówki lub początku wyrazu, końcówki poprzedniego słowa.

Niestety, język jest zbyt bogaty, aby uwzględnić wszystkie możliwe konteksty i wszystkie możliwe słowa. Z tego powodu tworzy się pewne abstrakcje wszystkich możliwych informacji z kontekstu, które pozwolą na wybór odpowiedniego tagu. Dla przykładu trigramowy ukryty model Markowa (HMM, ang. *Hidden Markov Model*) bierze pod uwagę tylko dwa poprzednie tagi jako kontekst. Abstrakcja kontekstu dla tego modelu jest uboga, a i tak tager uzyskuje dość dużą skuteczność. Od razu nasuwa się pytanie, czy nie dałoby się rozszerzyć zbioru informacji, na podstawie której tager podejmuje decyzje? Odpowiedź na to pytanie jest twierdząca.

Model maksimum entropii (w dalszej części będziemy używali skrótu MaxEnt) jest próbą stworzenia jak najbardziej ogólnego schematu postępowania w takim przypadku. MaxEnt nie definiuje abstrakcji kontekstu — pozwala na samodzielne zdefiniowanie cech kontekstowych, które posłużą do uczenia modelu. Takie podejście ma swoje zalety (duża ogólność), jak i wady (wybór cech kontekstowych jest najtrudniejszym zadaniem w konstrukcji modelu). Budowa modelu opiera się na następującym pomysłe: założmy, że cechy kontekstowe zostały dobrze wybrane przez konstruktora modelu i występują one w próbce treningowej tak samo często, jak w rzeczywistości. Przy takim założeniu otrzymujemy zbiór tagerów, z którego wybieramy ten najbardziej ogólny, który nie nakłada żadnych dodatkowych warunków.

Poniższa praca magisterska jest pierwszym podejściem do zastosowania metody maksimum entropii dla języka polskiego. Zbudowany model uzyskał poprawność tagowania na poziomie 93,1%. Wydaje się, że ten wynik można znacząco polepszyć, ponieważ użyty model oferuje dużą swobodę w doborze cech kontekstowych. Metoda maksimum entropii została z sukcesem zastosowana przez Ratnaparkiego do tagowania korpusu *Penn Treebank*. Podaje

on w swojej pracy [Ratnaparkhi 1996] skuteczność tagera MaxEnt na poziomie 96,6%, jednak tagowanie języka angielskiego jest uznawane za zadanie łatwiejsze niż tagowanie języków fleksyjnych, do których zalicza się język polski. Wynika to z dużo większej liczby możliwych tagów oraz większej przeciętnej liczby możliwych interpretacji przypadających na słowo w języku polskim. W języku angielskim wyróżnia się, w zależności od przyjętego tagsetu, od 35 do 130 tagów. Dla języka polskiego ten zbiór jest znacznie większy. W próbce Narodowego Korpusu Języka Polskiego wielkości 1,2 mln słów wystąpiło 1150 spośród 4179 potencjalnych możliwości.

W dalszej części pracy podjąłem próbę zdefiniowania wszystkich niestandardowych pojęć (m.in. entropii, analizatora morfologicznego) związanych z tagerem maksimum entropii, przedstawienia związanych z nimi intuicji i przykładów. Rozdział 2 przedstawia teoretyczne podejście do modelu maksimum entropii i ma charakter bardziej ścisły niż pozostałe. Do pełnego zrozumienia rozdziału 2 potrzebny jest podstawowa wiedza z zakresu analizy matematycznej i rachunku prawdopodobieństwa. Pozostałe rozdziały mają bardziej opisowy charakter.

Rozdział 1

Wstęp

We wstępie znajduje się krótkie wprowadzenie do problemu tagowania tekstów z Narodowego Korpusu Języka Polskiego. Zawiera definicje najważniejszych pojęć (m.in. tagu, leksemu, analizatora składniowego) oraz odpowiednie przykłady.

1.1. Zadanie tagowania

Zadanie tagowania jest jednym z podstawowych zadań przetwarzania języka naturalnego. Polega na przyporządkowaniu słowom w zdaniu tzw. tagów. Tag to etykieta nadawana słowu, która składa się ze zbioru par (atrybut, wartość) zawierającego pary typu:

- część mowy \rightarrow rzeczownik,
- liczba \rightarrow mnoga,
- przypadek \rightarrow mianownik,
- rodzaj \rightarrow żeński.

Część mowy będziemy oznaczali jako **POS** (*part-of-speech*). Pozostałe atrybuty to tzw. **atrybuty morfologiczne** albo **kategorie morfologiczne**. Liczba kategorii morfologicznych dla danego słowa zależy od tego, jaką część mowy reprezentuje tagowane słowo. Czasownik posiada osobę (np. *ja poszedłem, ty poszedłeś*), rzeczownik osoby nie posiada. Pełne zestawienie atrybutów dla poszczególnych części mowy można znaleźć w podrozdziale 1.3.

Istnieje wiele teorii na temat tego, ile słowo powinno mieć atrybutów oraz jakie powinny być zbiory wartości dla atrybutów. Tager przedstawiony w tej pracy został oparty na zbiorze tagów **NKJP Tagset**, który został użyty do tagowania Narodowego Korpusu Języka Polskiego. Tagset NKJP wyróżnia 35 części mowy oraz 13 kategorii morfologicznych. Więcej informacji na temat korpusu i użytego tagsetu znajduje się w podrozdziale 1.2.

Zadaniem **tagera** jest wybór najbardziej odpowiedniego tagu dla słowa w danym kontekście. Dla przykładu jako wejście tagera weźmy zdanie:

Skoro baba umiera, to pan doktor musi przyjść.

Tager powinien zwrócić nam powyższe zdanie z nadanymi odpowiednimi tagami. Przykład takiego wyniku przedstawiono w poniższej tabeli.

<i>Skoro</i>	conj
<i>baba</i>	subst:sg:nom:f
<i>umiera</i>	fin:sg:ter:imperf
<i>,</i>	interp
<i>to</i>	conj
<i>pan</i>	subst:sg:nom:m1
<i>doktor</i>	subst:sg:nom:m1
<i>musi</i>	fin:sg:ter:imperf
<i>przyjść</i>	inf:perf
<i>.</i>	interp

Słowo *baba* ma tag *subst:sg:nom:f*, który oznacza:

- subst → substantive (rzeczownik),
- sg → singular (liczba pojedyncza),
- nom → nominative (mianownik),
- f → feminine (rodzaj żeński).

Działanie tagera można podzielić na dwa etapy:

1. **Analiza morfologiczna**, czyli określenie dla danego słowa wszystkich form podstawowych tagowanego słowa oraz wszystkie możliwe tagi. Dla przykładu: słowo *mam* może mieć formę podstawową *mama*, *mamić* albo *mieć*. W zależności od tego, jaka jest poprawna forma podstawowa słowa *mam* w danym kontekście, będziemy mieli różne możliwe tagi. W procesie analizy morfologicznej nie uwzględnia się kontekstu, w którym wystąpiło dane słowo. W NKJP jako analizator morfologiczny został użyty Morfeusz. Więcej na jego temat można przeczytać w podrozdziale 1.4.
2. **Ujednoznacznianie morfologiczne (dezambiguacja morfologiczna)**, czyli określanie na podstawie kontekstu, jaką formę realizuje dane wystąpienie słowa. Ujednoznacznianiem morfologicznym w naszym przypadku zajmuje się model maksimum entropii.

Dla czytelności i prostoty w dalszej części pracy model maksimum entropii użyty do ujednoznaczniania morfologicznego będziemy nazywali tagerem. Pamiętajmy jednak, że używa on Morfeusza jako analizatora składniowego.

Do budowy tagerów wykorzystywane są różne techniki. Najpopularniejsze są tagery korzystające z metod statystycznych. Potrzebują one próbki języka ręcznie otagowanej przez lingwistów, na podstawie której budowany jest model wykorzystywany później do tagowania tekstów. Do takich tagerów zaliczamy m.in.

- drzewa decyzyjne,
- modele n-gramowe,
- tager HMM (korzystający z ukrytych modeli Markowa),
- tager oparty na sieciach neuronowych,
- tager Brilla,
- tager maksimum entropii.

1.2. Korpus

Autorzy Narodowego Korpusu Języka Polskiego na stronie internetowej [NKJP] podają następującą definicję korpusu.

„Korpus językowy to zbiór tekstów, w którym szukamy typowych użycí słów i konstrukcji oraz innych informacji o ich znaczeniu i funkcji. Bez dostępu do korpusu nie da się dziś prowadzić badań językoznawczych, pisać słowników ani podręczników języków obcych, tworzyć wyszukiwarek uwzględniających polską odmianę, tłumaczy komputerowych ani innych programów zaawansowanej technologii językowej. Korpus jest niezbędny do pracy językoznawcom, ale korzystają zeń często także informatycy, historycy, bibliotekarze, badacze literatury i kultury oraz specjaliści z wielu innych dziedzin humanistycznych i informatycznych.

Narodowy Korpus Języka Polskiego jest wspólną inicjatywą Instytutu Podstaw Informatyki PAN (koordynator), Instytutu Języka Polskiego PAN, Wydawnictwa Naukowego PWN oraz Zakładu Językoznawstwa Komputerowego i Korpusowego Uniwersytetu Łódzkiego, zarejestrowaną jako projekt badawczy rozwojowy Ministerstwa Nauki i Szkolnictwa Wyższego.”

Obecnie Narodowy Korpus Języka Polskiego liczy już ponad miliard słów. Próbką około 1,2 miliona słów z tego korpusu została ręcznie otagowana i posłużyła do ewaluacji skuteczności tagera maksimum entropii. Wyniki ewaluacji można znaleźć w podrozdziale 4.1. Narodowy Korpus Języka Polskiego korzysta z Tagsetu NKJP opisanego w podrozdziale 1.3.

1.3. Tagset

Tagset NKJP wyróżnia 35 części mowy **POS** oraz 11 kategorii morfologicznych **ATTR**. Poniżej przedstawiono listę wszystkich kategorii morfologicznych oraz ich wartości.

[ATTR]

number	= sg pl
case	= nom gen dat acc inst loc voc
gender	= m1 m2 m3 f n
person	= pri sec ter
degree	= pos com sup
aspect	= imperf perf
negation	= aff neg
accommodability	= congr rec
accentability	= akc nakc
post-prepositionality	= npraep praep
agglutination	= agl nagl
vocalicity	= nwok wok
fullstoppedness	= pun npun

Każdej części mowy przypisywane są odpowiednie atrybuty morfologiczne. Pełną zależność przedstawia poniższy spis. Zależność *adv* = [degree] oznacza, że stopień jest opcjonalnym atrybutem dla przysłówka. Dla przykładu każdy rzeczownik musi posiadać atrybuty *number:case:gender*, gdzie np. *number* jedną z możliwych wartości dla tej kategorii, to jest *sg* albo *pl*. Stąd pełnym tagiem dla słowa *lampę* powinien być *subst:sg:acc:f*.

[POS]

adja	=
adjp	=
adjc	=
conj	=
comp	=
interp	=
pred	=
xxx	=
adv	= [degree]
imps	= aspect
inf	= aspect
pant	= aspect
pcon	= aspect
qub	= [vocalicity]
prep	= case [vocalicity]
siebie	= case
subst	= number case gender
depr	= number case gender
ger	= number case gender aspect negation
ppron12	= number case gender person [accentability]
ppron3	= number case gender person [accentability] [post-prepositionality]
num	= number case gender accommodability
numcol	= number case gender accommodability
adj	= number case gender degree
pact	= number case gender aspect negation
ppas	= number case gender aspect negation
winien	= number gender aspect
praet	= number gender aspect [agglutination]
bedzie	= number person aspect
fin	= number person aspect
impt	= number person aspect
aglt	= number person aspect vocalicity
brev	= fullstoppedness
burk	=
interj	=

Opis zdefiniowanych części mowy oraz wykorzystanych atrybutów morfologicznych można znaleźć w pracy [Przepiórkowski 2009].

1.4. Analizator morfologiczny Morfeusz

Narodowy Korpus Języka Polskiego korzysta z Morfeusza — analizatora morfologicznego autorstwa Marcina Wolińskiego. Poniższy opis działania analizatora morfologicznego, przedstawione definicje oraz przykład pochodzą ze strony internetowej [Morfeusz].

Słowem nazywamy ciąg znaków w tekście w języku naturalnym zwykle wydzielony odstępami lub znakami interpunkcyjnymi. **Leksem** to abstrakcyjna jednostka języka, wyraz

słownikowy. **Formą wyrazową** nazywamy słowo zinterpretowane — przypisane do konkretnego leksemu i opisanie co do jego funkcji gramatycznej.

Analiza morfologiczna polega na określeniu dla danego słowa wszystkich form wszystkich leksemów, których może ono być wykładnikiem. W procesie tym nie uwzględnia się kontekstu, w którym wystąpiło dane słowo. W językoznawstwie termin analiza morfologiczna odnosi się raczej do rozkładania wyrazów na elementarne składniki morfologiczne (morfemy), być może sensowniej byłoby więc mówić o **analizie fleksyjnej**, niestety wydaje się, że to ten pierwszy termin utarł się w środowisku językoznawstwa komputerowego.

Ujednoznacznianiem morfologicznym (dezambiguacją morfologiczną) nazywamy określanie na podstawie kontekstu, jaką formę realizuje dane wystąpienie słowa. Program **Morfeusz** wykonuje analizę morfologiczną dla języka polskiego. W obecnej wersji nie zawiera modułu zgadującego nieznane słowa (można więc powiedzieć, że jest słownikiem morfologicznym).

Poniżej znajduje się przykład wyniku działania programu dla tekstu „Mam próbkę analizy morfologicznej.”:

Wyraz	Leksemy	Tagi dla leksemu
Mam	mama	subst:pl:gen:f
	mamić	impt:sg:sec:imperf
	mieć	fin:sg:pri:imperf
próbkę	próbka	subst:sg:acc:f
analizy	analiza	subst:sg:gen:f
		subst:pl:nom:f
		subst:pl:acc:f
		subst:pl:voc:f
morfologicznej	morfologiczny	adj:sg:gen.dat.loc:f:pos
		adj:sg:gen:f:pos
		adj:sg:dat:f:pos
		adj:sg:loc:f:pos
.	.	interp

Więcej na temat Morfeusza można przeczytać na stronie internetowej [Morfeusz] oraz w pracy [Woliński 2006].

1.5. Krótko o modelu maksimum entropii

Wyjściem analizatora morfologicznego jest zbiór możliwych tagów $T = \{t_1, \dots, t_k\}$ dla słowa s . Zadaniem tagera jest wybranie na podstawie kontekstu k takiego tagu t_i , dla którego $p(t_i|s, k)$, czyli prawdopodobieństwo wystąpienia tagu t w kontekście k jest maksymalne. Zatem tager ma do dyspozycji kontekst słowa, którego analizator morfologiczny nie używa. Z powyższej definicji wynika, że tak naprawdę model tagera jest pewnym warunkowym rozkładem prawdopodobieństwa. Pomysł na model maksimum entropii jest następujący:

1. Weźmy zbiór \mathcal{G} rozkładów prawdopodobieństwa, które mogą nadawać się na dobry tager.
2. Wybierzmy ze zbioru \mathcal{G} rozkład o największej entropii.

Szczegóły teoretyczne dotyczące wyboru odpowiedniego modelu zostały zawarte w następnym rozdziale.

Rozdział 2

Model maksimum entropii od strony teoretycznej

2.1. Entropia

Definicja 2.1.1. *Entropią* rozkładu prawdopodobieństwa $p = \{p_k\}_{k \in \{1, \dots, n\}}$ nazywamy wielkość:

$$H_t(p) = - \sum_{i=1}^n p_i \log_t(p_i). \quad (2.1)$$

Własności entropii:

- $H_t(p) \geq 0$,
- $H_t(p) \leq \log_t(n)$,
- $H_t(p)$ przyjmuje maksimum jeśli $p_k = \frac{1}{n}$ dla pewnego $k \in \{1, \dots, n\}$.

Entropia wyraża średnią ilość informacji przypadającej na jedno zdarzenie losowe. We wzorze (2.1) zwykle przyjmuje się jako podstawę logarytmu $t = 2$ lub $t = e$. Przyjmijmy konwencję $H(p) := H_2(p)$. Zmieniając podstawę logarytmu, mnożymy entropię wszystkich rozkładów przez stałą. Jeżeli zdarzenie losowe jest pewne, tzn. $\exists_k p_k = 1$, wtedy znajomość eksperymentu losowego nie wnosi żadnej nowej informacji — wynik eksperymentu znaleźliśmy jeszcze przed jego wykonaniem. W takim przypadku entropia $H_t(p)$ wynosi 0. Wprowadźmy jeszcze dwie definicje, a następnie podamy fakt, który pozwala nabrać odrobinę intuicji, czym tak naprawdę jest entropia.

Definicja 2.1.2. Niech $p = \{p_k\}_{k \in \{1, \dots, n\}}$ będzie dowolnym rozkładem prawdopodobieństwa. **Kodem** rozkładu p nazywamy odwzorowanie $\varphi : p \rightarrow \{0, 1\}^*$, które dla każdego $k \in \mathbb{N}$ oraz dowolnych $p^i \in p$ pozwala z ciągu $\varphi(p^1)\varphi(p^2)\dots\varphi(p^k)$ jednoznacznie odtworzyć ciąg p^1, p^2, \dots, p^k .

Definicja 2.1.3. Dla danego kodu φ , *średnią długość kodu* φ definiujemy jako:

$$L(\varphi) = \sum_{i=1}^n p_i \cdot |\varphi(p_i)|, \quad (2.2)$$

gdzie $|\varphi(p_i)|$ jest długością ciągu $\{0, 1\}^*$ przypisanego zdarzeniu p_i przez kod φ .

Fakt 2.1.4. Dla dowolnego kodu φ i rozkładu prawdopodobieństwa p zachodzi nierówność:

$$H(p) \leq L(\varphi). \quad (2.3)$$

Okazuje się jednak, że optymalny kod binarny dla p jest bardzo bliski wartości $H_2(p)$. Nierówności (2.3) nie da się poprawić.

Fakt 2.1.5. Dla każdego skończonego rozkładu prawdopodobieństwa p istnieje kod $\varphi : p \rightarrow \{0, 1\}^*$ spełniający nierówność:

$$H(p) \leq L(\varphi) \leq H(p) + 1. \quad (2.4)$$

Równość $L(\varphi) = H(p) + 1$ zachodzi tylko dla rozkładów, gdzie $p_k = 1$ dla pewnego k .

Powyższy fakt możemy interpretować następująco. Jeżeli chcemy w optymalny sposób zakodować nieskończony ciąg niezależnych zdarzeń p^1, p^2, \dots z rozkładu p , to średnio na zapis jednego zdarzenia p^k musimy zużyć co najmniej $H(p)$ bitów.

Definicja 2.1.6. Niech $p(x, y)$, $x \in X$, $y \in Y$, będzie rozkładem prawdopodobieństwa pary zmiennych losowych (X, Y) . Entropią warunkową $H_t(p_{Y|X})$ nazywamy wielkość:

$$H_t(p_{Y|X}) = - \sum_{x,y} p(x, y) \log_t p(y|x). \quad (2.5)$$

Entropię warunkową interpretujemy jako miarę losowości Y pod warunkiem, że znamy wartość X . Równość $H_t(p_X) = H_t(p_{X|Y})$ zachodzi wtedy i tylko wtedy, gdy X i Y są niezależnymi zmiennymi losowymi.

Własności warunkowej entropii:

- $H_t(p_{Y|X}) \geq 0$,
- $H_t(p_{Y|X}) \leq \log_t(n)$,
- $H_t(p_{Y|X})$ przyjmuje maksimum, jeśli $p_k = \frac{1}{n}$ dla pewnego $k = 1, \dots, n$.

W dalszej części będziemy opuszczać parametr t , przyjmując podstawę logarytmu równą 2, tzn. $H(p_{Y|X}) := H_2(p_{Y|X})$. Ponadto w dalszej części będziemy rozważali tylko entropię warunkową rozkładów warunkowych. Jeżeli z kontekstu będzie jasno wynikało, że p jest rozkładem warunkowym to będziemy pisać $H(p)$ zamiast $H(p_{Y|X})$.

2.2. Intuicje dotyczące zasady maksimum entropii

Zasada maksimum entropii brzmi następująco: spośród akceptowalnych rozkładów prawdopodobieństwa wybierz rozkład o największej entropii. Przedstawimy powyższą zasadę na prostym przykładzie. Przypuśćmy, że chcemy otagować słowo XYZ . Mamy do wyboru 32 tagi: rzeczownik (subst), przymiotnik (adj), liczebnik (num), imiesłów (praet), przyimek (prep), zaimek (ppron), spójnik (conj) itd. i żadnej informacji na temat rozkładu tagów dla słowa XYZ . Jeżeli nie wiemy nic więcej na temat rozkładu tagów dla słowa XYZ , to wybieramy rozkład o największej entropii — w tym przypadku rozkład jednostajny.

subst	adj	num	praet	prep	ppron	conj	...
$\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{32}$...

Przypuśćmy, że wiemy, iż słowo XYZ może być tylko rzeczownikiem, przymiotnikiem, liczebnikiem albo imiesłowem. Nie mając informacji o tym, jak rozkładają się prawdopodobieństwa między tymi czterema tagami, nie możemy żadnego z nich wyróżnić i dlatego wszystkim przypisujemy równe prawdopodobieństwa. Taki rozkład maksymalizuje entropię przy zadanych ograniczeniach.

subst	adj	num	praet	prep	ppron	conj	...
$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	...

Jeżeli wiemy, że słowo XYZ w 80% przypadków występuje jako przymiotnik albo liczebnik, to możemy dokładniej określić rozkład. Maksymalizując entropię, otrzymujemy następujący rozkład:

subst	adj	num	praet	prep	ppron	conj	...
$\frac{1}{10}$	$\frac{4}{10}$	$\frac{4}{10}$	$\frac{1}{10}$	0	0	0	...

Jeżeli dodatkowo dołożymy ograniczenie, że słowo XYZ jest rzeczownikiem z prawdopodobieństwem $\frac{1}{20}$, to wybieramy następujący rozkład:

subst	adj	num	praet	prep	ppron	conj	...
$\frac{1}{20}$	$\frac{4}{10}$	$\frac{4}{10}$	$\frac{3}{20}$	0	0	0	...

2.3. Model tagera

Naszym celem jest zbudowanie modelu stochastycznego, który dla zadanego kontekstu k dla tagowanego słowa oraz tagu t poda $p(t|k)$, tzn. prawdopodobieństwo tego, że tag t jest poprawny w kontekście k . Poprzez kontekst k rozumiemy całą historię dostępną w momencie tagowania słowa, tzn. cały tagowany tekst (zawierający tagowane słowo oraz słowa go poprzedzające i po nim następujące) oraz wcześniej nadane tagi. Zakładamy przy tym, że tagi są przypisywane słowom sekwencyjnie od lewej do prawej.

Musimy znaleźć sposób, aby oszacować $p(t|k)$, mając do dyspozycji zbiór treningowy $(k_1, t_1), (k_2, t_2), \dots, (k_N, t_N)$. Zbiorem treningowym jest zbiór kontekstów oraz przypisane słowom w tych kontekstach tagi, czyli otagowany korpus, na którym będziemy uczyć nasz model.

Definicja 2.3.1. *Oznaczmy przez \tilde{p} empiryczny rozkład prawdopodobieństwa na zbiorze treningowym:*

$$\tilde{p}(k, t) = \frac{1}{N} \times \text{liczba razy para } (k, t) \text{ wystąpiła w zbiorze treningowym.}$$

2.4. Cechy kontekstowe

Chcemy zbudować model statystyczny, który przy użyciu prawdopodobieństwa empirycznego \tilde{p} na zbiorze treningowym pozwoli nam oszacować prawdopodobieństwa $p(t|k)$. W podrozdziale 2.2 pokazywaliśmy jak wprowadzanie ograniczeń na rozkład wpływa na wybór rozkładu o największej entropii. Tam statystyki dla słowa XYZ były niezależne od kontekstu, jednak możemy również używać statystyk kontekstowych. Wcześniej używaliśmy entropii zwykłej, teraz będziemy używać entropii warunkowej dla rozkładu warunkowego. Zasada maksymalizacji entropii warunkowej jest identyczna jak zasada maksymalizacji standardowej entropii. Wprowadźmy teraz definicję funkcji cechy kontekstowej.

Definicja 2.4.1. Cechą kontekstową f nazywamy funkcję $f : \mathcal{K} \times \mathcal{T} \rightarrow \{0, 1\}$, gdzie \mathcal{K} jest zbiorem wszystkich możliwych kontekstów oraz \mathcal{T} jest zbiorem wszystkich dostępnych tagów.

Przykładem cechy kontekstowej może być np.

$$f(k, t) = \begin{cases} 1 & \text{jeżeli } t = \text{„rzeczownik” i } k(\text{słowo}) = \text{„stół”} \\ 0 & \text{wpp.} \end{cases} \quad (2.6)$$

Wartość oczekiwaną cechy kontekstowej f względem rozkładu empirycznego \tilde{p} będziemy oznaczać następująco:

$$\tilde{p}(f) \equiv \sum_{k,t} \tilde{p}(k, t) f(k, t). \quad (2.7)$$

Wartość oczekiwana cechy kontekstowej f względem rozkładu modelowego p wyraża się wzorem:

$$\sum_{k,t} p(k, t) f(k, t), \quad (2.8)$$

jednak my będziemy stosować wzór przybliżony:

$$p(f) \equiv \sum_{k,t} \tilde{p}(k) p(t|k) f(k, t). \quad (2.9)$$

Zauważmy, że stosujemy tutaj przybliżenie $p(k) \approx \tilde{p}(k)$. Podobne przybliżenie będziemy stosować w dalszej części pracy dla wzoru (2.5) na entropię warunkową:

$$H(p) = \sum_{k,t} p(k, t) \log p(t|k) \approx \sum_{k,t} \tilde{p}(k) p(t|k) \log p(t|k). \quad (2.10)$$

Wynika to z faktu, że zbiór wszystkich kontekstów \mathcal{K} jest ogromny, a modelowanie $p(k, t)$ skomplikowane, dlatego skupiamy się na modelowaniu prawdopodobieństwa warunkowego $p(t|k)$. Zakładamy również, że rozkład empiryczny brzegowy $\tilde{p}(k)$ jest wiarygodny i dobrze odzwierciedla prawdziwy rozkład kontekstów w języku. Zatem nasz model jest wyznaczony jednoznacznie przez rozkład warunkowy $p(t|k)$ dla $t \in \mathcal{T}$, $k \in \mathcal{K}$.

Definicja 2.4.2. Symbolem \mathcal{P} oznaczamy zbiór wszystkich warunkowych rozkładów prawdopodobieństwa postaci $p(t|k)$ dla $t \in \mathcal{T}$, $k \in \mathcal{K}$. Model tagera jest z definicji elementem przestrzeni \mathcal{P} .

Zależy nam na tym, aby wybrać takie cechy charakterystyczne f_i dla języka polskiego, aby na ich podstawie dało się poprawnie wybrać tag dla dowolnego słowa w dowolnym kontekście. Takie cechy mogą wyglądać bardzo różnie, ich konstrukcję dokładnie omawiamy w rozdziale 3. Cechy nie mogą być za bardzo skomplikowane, ponieważ wtedy napotykamy na problem rzadkości danych. Muszą występować względnie często w języku i w zbiorze treningowym. Chcielibyśmy wybrać taki model, aby wartość oczekiwana każdej cechy kontekstowej f_i względem rozkładu empirycznego \tilde{p} i rozkładu modelowego p była taka sama, tzn.

$$p(f_i) = \tilde{p}(f_i) \quad (2.11)$$

dla każdej wybranej cechy kontekstowej f_i . Równość (2.11) ogranicza nam klasę możliwych rozkładów prawdopodobieństw dla konstruowanego modelu.

2.5. Zastosowanie zasady maksimum entropii do wyboru tagera

Przypuśćmy, że mamy dane n cech kontekstowych f_i , które są istotne w modelowaniu rozkładu prawdopodobieństwa $p(t|k)$ dla naszego tagera. Chcielibyśmy wybrać rozkład zgodny z wybranymi cechami oraz warunkami (2.11).

Definicja 2.5.1. Niech $\mathcal{G} \subset \mathcal{P}$ oznacza podzbiór rozkładów warunkowych zgodnych z warunkami (2.11), tzn.

$$\mathcal{G} \equiv \left\{ p \in \mathcal{P} \mid p(f_i) = \tilde{p}(f_i) \text{ dla } i \in \{1, 2, \dots, n\} \right\}. \quad (2.12)$$

Zasada maksimum entropii każe wybrać nam taki rozkład $p \in \mathcal{G}$, który ma maksymalną warunkową entropię (zobacz definicję 2.5.2).

Definicja 2.5.2 (Zasada maksimum entropii). Wybierając rozkład warunkowy p spośród zbioru \mathcal{G} dostępnych warunkowych rozkładów prawdopodobieństwa, zawsze wybieramy $p^* \in \mathcal{G}$ o największej warunkowej entropii:

$$p^* = \arg \max_{p \in \mathcal{G}} H(p). \quad (2.13)$$

Zastosowanie zasady maksimum entropii z definicji 2.5.2 sprowadza się do rozwiązania zadania optymalizacji funkcji $H(p)$ w zbiorze \mathcal{G} . W prostych przypadkach możemy znaleźć rozwiązanie w postaci analitycznej, jednak w przedstawionej tutaj ogólnej postaci nie da się tego problemu rozwiązać. Do znalezienia rozkładu maksymalizującego entropię użyjemy metody mnożników Lagrange’a. Kolejne kroki obliczeń przedstawiamy poniżej. Szczegółowy opis metody mnożników Lagrange’a znajduje się w dodatku A, tutaj przedstawiamy tylko jej zastosowanie do maksymalizacji entropii. Funkcja $H(p)$ jest wklęsła i różniczkowalna, a zbiór \mathcal{G} został określony za pomocą równań liniowych ze względu na p — zobacz równania (2.10) oraz (2.12). W związku z tym możemy swobodnie korzystać z metody Lagrange’a.

1. Oryginalnym zadaniem optymalizacji jest problem znalezienia $p^* \in \mathcal{G}$ spełniającego równanie:

$$p^* = \arg \max_{p \in \mathcal{G}} H(p). \quad (2.14)$$

Inaczej mówiąc, poszukujemy takiego p , które maksymalizuje $H(p)$ oraz spełnia następujące warunki:

$$p(t|k) \geq 0 \text{ dla każdego } t \in \mathcal{T}, k \in \mathcal{K}, \quad (2.15)$$

$$\sum_t p(t|k) = 1 \text{ dla każdego } k \in \mathcal{K}, \quad (2.16)$$

$$\tilde{p}(f_i) = p(f_i), \text{ czyli } \sum_{k,t} \tilde{p}(k, t) f_i(k, t) = \sum_{k,t} \tilde{p}(k) p(t|k) f_i(k, t). \quad (2.17)$$

Warunki (2.15) i (2.16) gwarantują, że p jest warunkowym rozkładem prawdopodobieństwa.

2. Dla każdej cechy f_i spełniającej równanie $\tilde{p}(f_i) = p(f_i)$ wprowadzamy mnożnik Lagrange’a λ_i , a dla warunku (2.16) wprowadzamy γ . Definiujemy lagranżjan $\Lambda(p, \lambda, \gamma)$ następująco:

$$\Lambda(p, \lambda, \gamma) \equiv H(p) + \sum_i \lambda_i [p(f_i) - \tilde{p}(f_i)] + \gamma [\sum_t p(t|k) - 1]. \quad (2.18)$$

Parametry $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ oraz γ odpowiadają $n + 1$ równościom, nakładającym ograniczenia na zbiór, w którym szukamy maksimum funkcji $H(p)$.

3. Ustalamy γ , λ i maksymalizujemy funkcję $\Lambda(p, \lambda, \gamma)$ z równania (2.18) względem p . Prowadzi to do ustalenia parametru $p = p_{\lambda, \gamma}$. Traktujemy teraz funkcję $\Lambda(p_{\lambda, \gamma}, \lambda, \gamma)$ jako funkcję od parametrów λ i γ :

$$p_{\lambda, \gamma} \equiv \arg \max_{p \in \mathcal{P}} \Lambda(p, \lambda, \gamma), \quad (2.19)$$

$$\Psi(\lambda, \gamma) \equiv \Lambda(p_{\lambda, \gamma}, \lambda, \gamma). \quad (2.20)$$

Problem minimalizacji funkcji Ψ względem rzeczywistych parametrów λ, γ nazywamy problemem dualnym do problemu (2.14), a funkcję Ψ funkcją dualną do $H(p)$.

4. Wartość $p_{\lambda, \gamma}$ możemy policzyć wprost. W tym celu znajdziemy punkty, w których funkcja Λ ma zerowe pochodne cząstkowe po $p(t|k)$. Przypomnijmy, że:

$$\begin{aligned} \Lambda(p, \lambda, \gamma) &= H(p) + \sum_i \lambda_i \left[p(f_i) - \tilde{p}(f_i) \right] + \gamma \left[\sum_t p(t|k) - 1 \right] \\ &= - \sum_{k,t} \tilde{p}(k) p(t|k) \log p(t|k) + \sum_i \lambda_i \left[\sum_{k,t} \tilde{p}(k) f_i(k, t) (p(t|k) - \tilde{p}(t|k)) \right] \\ &\quad + \gamma \left[\sum_t p(t|k) - 1 \right]. \end{aligned} \quad (2.21)$$

Ustalmy λ i γ , policzmy pochodne cząstkowe po $p(t|k)$ i przyrównajmy do zera:

$$\frac{\partial \Lambda}{\partial p(t|k)} = -\tilde{p}(k) [1 + \log p(t|k)] - \sum_i \lambda_i \tilde{p}(k) f_i(k, t) + \gamma = 0. \quad (2.22)$$

Z układu równań różniczkowych powstałego z (2.22) dla wszystkich par $(t, k) \in \mathcal{T} \times \mathcal{K}$ możemy łatwo wyznaczyć $p_{\lambda, \gamma}(t|k)$:

$$p_{\lambda, \gamma}(t|k) = \exp \left(\sum_{i=1}^n \lambda_i f_i(k, t) \right) \exp \left(-\frac{\gamma}{\tilde{p}(k)} - 1 \right). \quad (2.23)$$

Zauważmy, że czynnik $\exp \left(-\frac{\gamma}{\tilde{p}(k)} - 1 \right)$ wziął się z warunku $\sum_t p_{\lambda, \gamma}(t|k) = 1$, zatem możemy przepisać równanie (2.23) do następującej postaci:

$$p_{\lambda, \gamma}(t|k) = \frac{1}{Z_{\lambda}(k)} \exp \left(\sum_{i=1}^n \lambda_i f_i(k, t) \right), \quad (2.24)$$

gdzie $Z_{\lambda}(k)$ jest czynnikiem normalizacyjnym dobranym tak, by spełnione było równanie $\sum_t p_{\lambda, \gamma}(t|k) = 1$, czyli:

$$Z_{\lambda}(k) = \sum_{t \in \mathcal{T}} \exp \left(\sum_i \lambda_i f_i(k, t) \right). \quad (2.25)$$

Wtedy $p_{\lambda, \gamma}$ nie zależy od γ , więc możemy pisać $p_{\lambda, \gamma} \equiv p_{\lambda}$.

5. Podstawiając (2.24) do (2.21) oraz korzystając z (2.16), możemy funkcję $\Psi(\lambda, \gamma)$ wyrazić następująco (dla czytelności przyjmujemy $p \equiv p_\lambda$):

$$\begin{aligned}
\Psi(\lambda, \gamma) &= \Lambda(p, \lambda, \gamma) \\
&= - \sum_{k,t} \tilde{p}(k) p(t|k) \log p(t|k) + \sum_i \lambda_i \left[\sum_{k,t} \tilde{p}(k) f_i(k, t) (p(t|k) - \tilde{p}(t|k)) \right] \\
&\quad + \gamma \left[\sum_t p(t|k) - 1 \right] \\
&\stackrel{(2.17)}{=} - \sum_{k,t} \tilde{p}(k) p(t|k) \log p(t|k) + \gamma \left[\sum_t p(t|k) - 1 \right] \\
&\stackrel{(2.16)}{=} - \sum_{k,t} \tilde{p}(k) p(t|k) \log p(t|k) \\
&\stackrel{(2.24)}{=} - \sum_{k,t} \left\{ \tilde{p}(k) \frac{1}{Z_\lambda(k)} \exp \left(\sum_i \lambda_i f_i(k, t) \right) \left[-\log Z_\lambda(k) + \sum_i \lambda_i f_i(k, t) \right] \right\} \\
&= \sum_{k,t} \left\{ \tilde{p}(k) \frac{1}{Z_\lambda(k)} \exp \left(\sum_i \lambda_i f_i(k, t) \right) \log Z_\lambda(k) \right\} \\
&\quad - \sum_{k,t} \left\{ \tilde{p}(k) \frac{1}{Z_\lambda(k)} \exp \left(\sum_i \lambda_i f_i(k, t) \right) \left[\sum_i \lambda_i f_i(k, t) \right] \right\} \\
&= \sum_k \left\{ \tilde{p}(k) \log Z_\lambda(k) \left[\frac{1}{Z_\lambda(k)} \sum_t \exp \left(\sum_i \lambda_i f_i(k, t) \right) \right] \right\} \\
&\quad - \sum_{k,t} \left\{ \tilde{p}(k) \frac{1}{Z_\lambda(k)} \exp \left(\sum_i \lambda_i f_i(k, t) \right) \left[\sum_i \lambda_i f_i(k, t) \right] \right\} \\
&\stackrel{(2.25)}{=} \sum_k \tilde{p}(k) \log Z_\lambda(k) - \sum_{k,t} \left\{ \tilde{p}(k) \frac{1}{Z_\lambda(k)} \exp \left(\sum_i \lambda_i f_i(k, t) \right) \left[\sum_i \lambda_i f_i(k, t) \right] \right\} \\
&\stackrel{(2.24)}{=} \sum_k \tilde{p}(k) \log Z_\lambda(k) - \sum_{k,t} \left\{ \tilde{p}(k) p(t|k) \left[\sum_i \lambda_i f_i(k, t) \right] \right\} \\
&= \sum_k \tilde{p}(k) \log Z_\lambda(k) - \sum_i \lambda_i \left\{ \sum_{k,t} \tilde{p}(k) p(t|k) f_i(k, t) \right\} \\
&\stackrel{(2.9)}{=} \sum_k \tilde{p}(k) \log Z_\lambda(k) - \sum_i \lambda_i p(f_i) \\
&\stackrel{(2.17)}{=} \sum_k \tilde{p}(k) \log Z_\lambda(k) - \sum_i \lambda_i \tilde{p}(f_i).
\end{aligned}$$

Zauważmy, że funkcja Ψ nie zależy od γ , zatem w dalszej części będziemy pisać $\Psi(\lambda)$.

6. Znajdujemy λ^* minimalizujące funkcję $\Psi(\lambda)$. Uwaga: zgodnie z metodą mnożników Lagrange'a w tym miejscu wykonujemy minimalizację. Zainteresowanych szczegółami tej dziwnej operacji odsyłam do dodatku A. Zatem:

$$\lambda^* = \arg \min_{\lambda} \Psi(\lambda). \quad (2.26)$$

Rozwiązanie tego równania nie wyraża się wzorem analitycznym. Minimalizację wykonujemy z użyciem gradientowej metody Newtona.

7. Znajdujemy p^* spełniające równanie (2.13), za pomocą równania:

$$p^* = p\lambda^*. \quad (2.27)$$

2.6. Zasada maksimum entropii a zasada największej wiarygodności

Okazuje się, że rozkład p^* maksymalizujący entropię i rozkład p_* maksymalizujący logarytmiczną funkcję wiarygodności są sobie równe.

Przypuśćmy, że na podstawie korpusu treningowego $(k_1, t_1), \dots, (k_N, t_N)$ wybraliśmy prawdopodobieństwo warunkowe $p = \{p(t|k)\}_{k \in \mathcal{K}, t \in \mathcal{T}}$ dla naszego modelu. Dla wybranego rozkładu p możemy policzyć wiarygodność zbudowanego modelu p , licząc prawdopodobieństwo wystąpienia w modelu korpusu treningowego (o rozkładzie empirycznym \tilde{p}). Innymi słowy jest to prawdopodobieństwo *a posteriori* zaobserwowanej próby. W naszym przypadku wiarygodność modelu wyraża się następującym wzorem:

$$l_{\tilde{p}}(p) \equiv \prod_{k \in \mathcal{K}, t \in \mathcal{T}} p(t|k)^{\tilde{p}(k,t)}. \quad (2.28)$$

Funkcję l nazywamy funkcję wiarygodności dla rozkładu p (ang. *likelihood function*). Zwykle definiuje się również funkcję $L \equiv \log l$ zwaną *log-likelihood function*, która przyjmuje prostszą postać:

$$L_{\tilde{p}}(p) \equiv \log l_{\tilde{p}}(p) = \sum_{k \in \mathcal{K}, t \in \mathcal{T}} \log p(t|k)^{\tilde{p}(k,t)} \equiv \sum_{k \in \mathcal{K}, t \in \mathcal{T}} \tilde{p}(k,t) \log p(t|k). \quad (2.29)$$

Definicja 2.6.1. *Rozkładem największej wiarygodności nazwiemy rozkład $p_* \in \mathcal{G}$ (zobacz definicję 2.5.1), który maksymalizuje funkcję $L_{\tilde{p}}(p)$, tzn.*

$$p_* = \arg \max_{p \in \mathcal{G}} L_{\tilde{p}}(p). \quad (2.30)$$

Twierdzenie 2.6.2. *Zachodzi równość:*

$$L_{\tilde{p}}(p_\lambda) = -\Psi(\lambda). \quad (2.31)$$

Dowód.

$$\begin{aligned} L_{\tilde{p}}(p_\lambda) &\stackrel{(2.29)}{=} \sum_{k,t} \tilde{p}(k,t) \log p(t|k) \\ &\stackrel{(2.24)}{=} \sum_{k,t} \tilde{p}(k,t) \log \left[\frac{1}{Z_\lambda(k)} \exp \left(\sum_{i=1}^n \lambda_i f_i(k,t) \right) \right] \\ &= \sum_{k,t} \tilde{p}(k,t) \left[-\log Z_\lambda(k) + \sum_{i=1}^n \lambda_i f_i(k,t) \right] \\ &= -\sum_k \log Z_\lambda(k) \left[\sum_t \tilde{p}(k,t) \right] + \sum_{i=1}^n \lambda_i \left[\sum_{k,t} \tilde{p}(k,t) f_i(k,t) \right] \\ &= -\sum_k \log Z_\lambda(k) \tilde{p}(k) + \sum_{i=1}^n \lambda_i \tilde{p}(f) \\ &= -\Psi(\lambda) \end{aligned} \quad (2.32)$$

□

Twierdzenie 2.6.3. *Rozkład maksymalizujący entropię jest jednocześnie rozkładem maksymalizującym funkcję wiarygodności danych treningowych w modelu, tzn. zachodzi równość:*

$$p_* = p^*. \quad (2.33)$$

Dowód.

$$\begin{aligned} p_* &\stackrel{(2.30)}{=} \arg \max_{p \in \mathcal{G}} L_{\tilde{p}}(p) \\ &= \arg \max_{\lambda} L_{\tilde{p}}(p_{\lambda}) \\ &= \arg \max_{\lambda} -\Psi(\lambda) \\ &= \arg \min_{\lambda} \Psi(\lambda) \\ &= p^* \end{aligned} \quad (2.34)$$

Wnioskujemy stąd, że rozkład p^* maksymalizujący entropię jest jednocześnie rozkładem maksymalizującym funkcję wiarygodności danych treningowych w modelu. \square

Uwaga 2.6.4. *Powyższy rozdział został w dużej mierze oparty na często cytowanej w literaturze pracy [Berger et al. 1996]. Praca zawiera sformułowania większości podanych wyżej twierdzeń, ale nie podaje szczegółowo wszystkich przejść. Ponadto w pracy znalazła się pomyłka, polegającą na zdefiniowaniu funkcji $\Psi(\lambda) = L(p_{\lambda}, \lambda)$, podczas gdy w dalszej dyskusji pojawia się $\Psi(\lambda) = -L(p_{\lambda}, \lambda)$. Tę pomyłkę zauważył już Łukasz Dębowski i opisał w pracy [Dębowski 2001].*

Rozdział 3

Wybór cech kontekstowych

Definicję cechy kontekstowej można znaleźć w podrozdziale 2.4, w tym rozdziale będziemy jej swobodnie używać. Intuicja podpowiada, że cecha kontekstowa przyjmująca w danym kontekście k wartość jeden wskazuje na to, że istnieją wyraźne przesłanki dla wyboru lub odrzucenia tagu t w kontekście k . Zatem fakt, że cecha kontekstowa f przyjmuje wartość jeden dla kontekstu k i tagu t może mieć pozytywny lub negatywny wpływ na wybór tagu t dla tego kontekstu.

Wybór cech kontekstowych jest najtrudniejszym elementem budowy modelu maksimum entropii. Cechy muszą być lingwistyczne uzasadnienie, tzn. muszą pomagać w wyborze odpowiedniego tagu. Cechy kontekstowe muszą być też dobrze reprezentowane w korpusie tak, by spełniona była równość $\tilde{p}(f) = p(f)$ — zobacz równania (2.7) i (2.9).

3.1. Dwuetapowy wybór tagu dla słowa

W badanej próbie korpusu NKJP wielkości 1,2 mln segmentów wystąpiło 968 spośród 3629 potencjalnych tagów z NKJP Tagset. Tak duża liczba tagów nie pozwala na zbudowanie jednego modelu maksimum entropii dla znajdowania odpowiedniego tagu dla słowa. Są ku temu trzy powody:

1. Ogromna złożoność obliczeniowa takiego modelu.
2. Niedokładność wynikająca z tego, że dla różnych części mowy różne wartości cechy kontekstowej wskazują na tę samą wartość kategorii morfologicznej. Dla przykładu weźmy słowo w rodzaju żeńskim oraz kategorię morfologiczną przypadku. Przymiotnik w dopełniaczu będzie zakończony na *-ej*, natomiast rzeczownik na *-i* (np. *małej dziewczynki*). Widzimy tutaj bardzo dokładną zależność między częścią mowy a wartością cechy decydującej o wyborze dopełniacza.
3. Różne części mowy posiadają różne zestawy atrybutów morfologicznych.

Z powyższych powodów wybór tagu dla kontekstu dokonujemy w dwóch krokach:

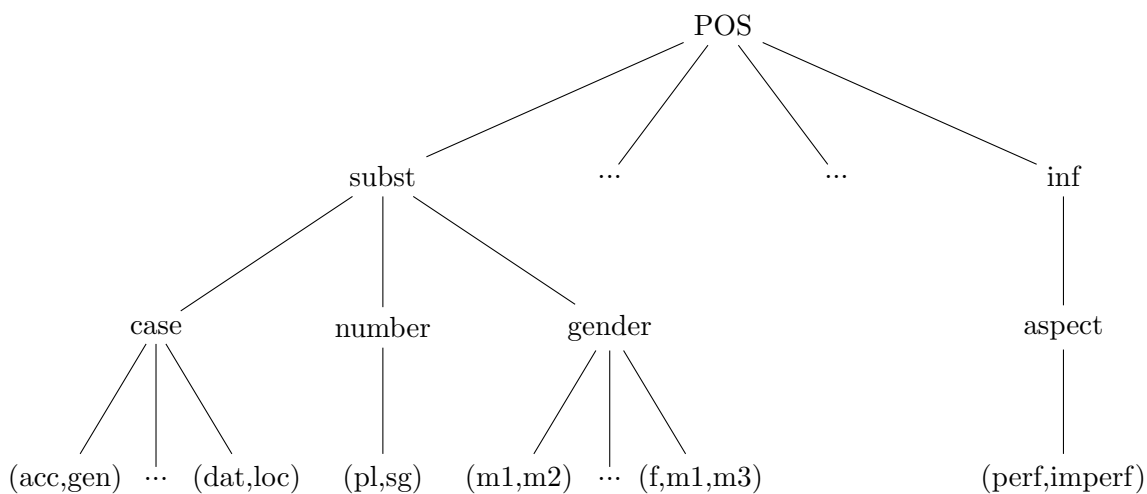
1. Wybór części mowy (np. rzeczownik, przymiotnik itd.).
2. Niezależny wybór wartości odpowiednich atrybutów morfologicznych (przypadka, rodzaju itd.) dla wcześniej wybranej części mowy.

3.2. Klasy niejednoznaczności

Analizator morfologiczny Morfeusz wybiera zbiór¹ wszystkich możliwych tagów (t_1, t_2, \dots, t_n) dla danego słowa s w kontekście k . Zadaniem tagera jest wybór spośród zbioru (t_1, t_2, \dots, t_n) jednego tagu t , najbardziej pasującego do kontekstu k . Niech (c_1, c_2, \dots, c_m) będzie zbiorem wszystkich części mowy, którymi może być słowo s . Zbiór (c_1, c_2, \dots, c_m) nazywamy **klasą niejednoznaczności** (ang. *ambiguity class*) części mowy i oznaczamy AC_{POS} . Dla każdej klasy niejednoznaczności części mowy (c_1, c_2, \dots, c_m) budujemy oddzielny model maksimum entropii, który będzie wybierał poprawną część mowy spośród (c_1, c_2, \dots, c_m) . W rozważanej próbkę Narodowego Korpusu Języka Polskiego o wielkości 1,2 mln segmentów występuje 390 różnych klas niejednoznaczności dla części mowy. Podczas konstruowania tagera odrzucamy wszystkie klasy niejednoznaczności, które wystąpiły mniej niż 20 razy. Daje to 223 klasy odrzucone, 167 pozostawionych. Dla pozostawionych klas budujemy niezależne modele maksimum entropii.

Przypuśćmy, że tager wskazał część mowy c jako najbardziej pasującą do kontekstu k . Następnym, po wyborze części mowy, krokiem jest wybór wartości atrybutów morfologicznych. Dla każdej kategorii morfologicznej tworzymy klasę niejednoznaczności dostępnych wartości pod warunkiem, że częścią mowy jest c . Dla przykładu: $AC_{CASE} = (acc, nom)$ to klasa niejednoznaczności przypadku składająca się z dwóch wartości.

Na 13 kategorii morfologicznych przypada 255 różnych klas niejednoznaczności, z czego 87 odrzucamy w naszym tagerze, ponieważ wystąpiły w korpusie mniej niż 10 razy. Zbudowany tager składa się z około 170 modeli POS (każdy dla innej klasy niejednoznaczności) do wyboru części mowy oraz około 170 modeli dla kategorii morfologicznych. Należy przy tym pamiętać, że np. modele dla klasy niejednoznaczności przypadku (acc, nom) dla rzeczownika i przymiotnika to dwa oddzielne modele. Modele maksimum entropii dla tagowania kategorii morfologicznych tworzą drzewiastą strukturę zależności, której część przedstawiona została poniżej. Liście odpowiadają odrębnym modelom maksimum entropii i są etykietowane klasami niejednoznaczności.



¹Zbiór oznaczamy okrągłymi nawiasami (t_1, t_2, \dots, t_n) zamiast $\{t_1, t_2, \dots, t_n\}$ z powodu przyjętych konwencji. Zakładamy również, że jest on uporządkowany alfabetycznie.

3.3. Przykład użycia modelu do wyboru tagu dla słowa

W celu pełnego zrozumienia działania tagera, warto prześledzić przykład użycia tagera w wybranym kontekście. Zobaczmy, jak odbywa się wybór części mowy i przypadku w uprzednio zbudowanym tagerze dla konkretnego słowa i kontekstu. Przykład został zaczerpnięty z korpusu NKJP, a przedstawione cechy kontekstowe z prawdziwego modelu. Weźmy następujące zdanie:

Człowiek to jeszcze jedna zaleta miasta .
subst pred qub (adj,subst,fin) (subst) (subst) (interp)

Przypuśćmy, że chcemy przypisać tag słowu *jedna* w podanym wyżej zdaniu. Tager dysponuje następującym kontekstem k_{pos} :

prevPrevPos → pred
prevPos → qub
prevWord → „jeszcze”
word → „jedna”
lastLetter → „a”
lastTwoLetters → „na”
lastThreeLetters → „dna”
nextPosAC → (subst)

Atrybut *nextPosAC* oznacza klasę niejednoznaczności dla części mowy następnego (w stosunku do tagowanego) słowa. W przypadku *nextPosAC* = (*subst*) analizator morfologiczny dopuszcza tylko jedną możliwą część mowy — rzeczownik. Dokładne schematy wszystkich cech kontekstowych można znaleźć w podrozdziale 3.4. Klasa niejednoznaczności dla części mowy została wybrana przez Morfeusza i jest nią w tym przypadku $AC_{POS} = (adj, fin, subst)$. Wybieramy model odpowiadający tej klasie AC_{POS} . W tej chwili spójrzmy tylko na aktywne cechy kontekstowe f_i i odpowiadające im wartości λ_i . Aktywne cechy kontekstowe to cechy, które przyjmują wartość jeden dla jednej z możliwych części mowy: *adj*, *fin* lub *subst*.

i	λ_i	cecha kontekstowa f_i
1	-9,027	prevPos=qub i t=subst
2	-1,851	nextPosAC=(subst) i t=subst
3	-3,697	lastLetter=„a” i t=fin
4	-1,624	word=„jedna” i t=subst
5	-1,624	lastTwoLetters=„na” i t=subst
6	-1,624	lastThreeLetters=„dna” i t=subst
7	0,711	lastTwoLetters=„na” t t=adj
8	0,711	word=„jedna” i t=adj

Przypomnijmy postać parametryczną $p(t|k_{pos})$ z równania (2.24):

$$p_{\lambda}(t|k_{pos}) = \frac{1}{Z_{\lambda(k_{pos})}} \exp \left(\sum_{i=1}^n \lambda_i f_i(k_{pos}, t) \right). \quad (3.1)$$

Zadaniem tagera jest wybór tagu $t \in (adj, fin, subst)$, który maksymalizuje $p(t|k_{pos})$. Podstawiając kolejno tagi ze zbioru (*adj*, *fin*, *subst*) do równania (3.1), otrzymujemy prawdopodobieństwa $p(adj|k_{pos})$, $p(fin|k_{pos})$ i $p(subst|k_{pos})$. Zauważmy, że w równaniu (3.1) wartość

$\frac{1}{Z_\lambda(k_{pos})}$ nie zależy od t . Ponadto \exp jest funkcją monotoniczną rosnącą, zatem wybór tagu sprowadza się do wyboru takiego t , dla którego wartość sumy $\sum_{i=1}^n \lambda_i f_i(k_{pos}, t)$ jest największa. Przeliczmy wartości tej sumy dla kolejnych tagów.

$$\begin{aligned}
p(adj|k_{pos}) &= \sum_{i=1}^8 \lambda_i f_i(k_{pos}, adj) \\
&= \lambda_7 f_7(k_{pos}, adj) + \lambda_8 f_8(k_{pos}, adj) \\
&= 0,711 + 0,711 = 1,422 \\
p(fin|k_{pos}) &= \sum_{i=1}^8 \lambda_i f_i(k_{pos}, fin) \\
&= \lambda_3 f_3(k_{pos}, fin) \\
&= -3,697 \\
p(subst|k_{pos}) &= \sum_{i=1}^8 \lambda_i f_i(k_{pos}, subst) \\
&= \lambda_1 f_1(k_{pos}, subst) + \lambda_2 f_2(k_{pos}, subst) + \lambda_4 f_4(k_{pos}, subst) \\
&\quad + \lambda_5 f_5(k_{pos}, subst) + \lambda_6 f_6(k_{pos}, subst) \\
&= -9,027 - 1,851 - 1,624 - 1,624 - 1,624 \\
&= -15,75
\end{aligned}$$

Z wykonanych obliczeń otrzymujemy nierówności:

$$p(adj|k_{pos}) > p(fin|k_{pos}) > p(subst|k_{pos}). \quad (3.2)$$

Zatem najbardziej prawdopodobną częścią mowy w danym kontekście jest przymiotnik (*adj*) i on zostaje wybrany przez tager.

Po wyborze części mowy zostaje nam wybór wartości kategorii morfologicznych: *number*, *case*, *gender* i *degree*. Zobaczmy, jak tager poradzi sobie z wyborem przypadku. Morfeusz przewidział dla formy przymiotnikowej *jedna* dwie wartości przypadku: mianownik i wołącz, czyli $AC_{CASE} = (nom, voc)$. Tager dysponuje następującym kontekstem k_{case} :

word	→	„jedna”
prevWord	→	„jeszcze”
prevCtag	→	qub
prevPrevCag	→	pred
nextCtagAC	→	(subst)
lastLetter	→	„a”
lastTwoLetters	→	„na”
lastThreeLetters	→	„dna”
prevCase	→	
prevPrevCase	→	
nextCaseAC	→	(nom)

Ponadto w trakcie trenowania modelu zostały wyuczone cechy f_i oraz odpowiadające im wartości λ_i . Cechy aktywne dla tagów *nom* lub *voc* w podanym kontekście są następujące:

i	λ_i	cecha kontekstowa f_i
1	-2,357	lastTwoLetters=„na” i t=voc
2	-1,579	lastLetter=„a” i t=voc
3	-1,159	nextCtagAC=(subst) i t=voc
4	0,939	word=„jedna” i t=voc
5	-0,706	lastThreeLetters=„dna” i t=voc
6	0,125	prevPrevCag=pred i t=nom
7	0,121	prevCtag=qub i t=nom
8	0,111	nextCaseAC=(nom) i t=nom
9	0,085	nextCtagAC=(subst) i t=nom
10	0,080	lastTwoLetters=„na” i t=nom
11	0,077	lastThreeLetters=„dna” i t=nom
12	0,069	word=„jedna” i t=nom

Chcemy wybrać takie $t \in (nom, voc)$, dla którego $p(t|k_{case})$ jest maksymalne. Porównajmy wartości $p(nom|k_{case})$ oraz $p(voc|k_{case})$ poprzez porównanie sum $\sum_{i=1}^n \lambda_i f_i(k_{case}, t)$ dla obu tagów.

$$\begin{aligned}
p(nom|k_{case}) &= \sum_{i=1}^{12} \lambda_i f_i(k_{case}, nom) \\
&= \lambda_6 f_6(k_{case}, nom) + \lambda_7 f_7(k_{case}, nom) + \lambda_8 f_8(k_{case}, nom) \\
&\quad + \lambda_9 f_9(k_{case}, nom) + \lambda_{10} f_{10}(k_{case}, nom) \\
&\quad + \lambda_{11} f_{11}(k_{case}, nom) + \lambda_{12} f_{12}(k_{case}, nom) \\
&= 0,125 + 0,121 + 0,111 + 0,085 + 0,080 + 0,077 + 0,069 \\
&= 0,668 \\
p(voc|k_{case}) &= \sum_{i=1}^{12} \lambda_i f_i(k_{case}, voc) \\
&= \lambda_1 f_1(k_{case}, voc) + \lambda_2 f_2(k_{case}, voc) + \lambda_3 f_3(k_{case}, voc) \\
&\quad + \lambda_4 f_4(k_{case}, voc) + \lambda_5 f_5(k_{case}, voc) \\
&= -2,357 - 1,579 - 1,159 + 0,939 - 0,706 \\
&= -4,862
\end{aligned}$$

Zatem tager wybrała mianownik jako najbardziej prawdopodobny przypadek. W przedstawionym zdaniu tager wybrał poprawnie zarówno część mowy, jak i przypadek.

3.4. Użyte w modelu schematy cech kontekstowych

Wybór cech kontekstowych dla modelu maksimum entropii jest najtrudniejszym etapem budowy tagera. Osiągnięta skuteczność 93,1% na poziomie pełnego tagu uzyskano, wykorzystując przedstawione poniżej schematy cech kontekstowych. Ze schematu cechy kontekstowej, jego wartości wynikającej z kontekstu i przewidywanego tagu tworzymy cechy kontekstowe. Pozycja słowa liczona jest względem tagowanego słowa, np. słowo na pozycji -1 to słowo poprzedzające tagowany wyraz. Dla przykładu:

Schemat cechy kontekstowej	→	słowo na pozycji 0
Wartość schematu w kontekście	→	słowo=„jedna”
Tag	→	adj
Cecha kontekstowa	→	$f(k, t) = \mathbb{1}_{\{\text{słowo}=\text{„jedna” i } t=\text{adj}\}}$

Wybór schematu cech kontekstowych odbywa się ręcznie. Budowa cech kontekstowych na podstawie schematów odbywa się automatycznie na etapie trenowania modelu. Pojedyncze modele maksimum entropii posiadają do 30 000 cech kontekstowych.

W podrozdziałach 3.4.1 i 3.4.2 przedstawiono użyte w tagerze schematy cech kontekstowych dla części mowy oraz poszczególnych kategorii morfologicznych. Cechy kontekstowe dla części mowy i wspólne dla kategorii morfologicznych zostały opatrzone odpowiednim przykładem ze zdania:

<i>Człowiek</i>	<i>to</i>	<i>jeszcze</i>	<i>jedna</i>	<i>zaleta</i>	<i>miasta</i>	.
subst	pred	qub	(adj,subst,fin)	(subst)	(subst)	(interp)
-3	-2	-1	0	1	2	3

W nawiasach podano klasy niejednoznaczności dla części mowy odpowiadające kolejnym słowom. Na samym dole są pozycje słów z kontekstu. Tagowane słowo ma pozycję 0, słowo je poprzedzające pozycję -1 itd.

3.4.1. Części mowy

Pozycja	Cecha	Przykładowa wartość
-2	część mowy	pred
-1	część mowy	qub
-1	słowo	jeszcze
0	słowo	jedna
0	ostatnia litera	a
0	ostatnie dwie litery	na
0	ostatnie trzy litery	dna
1	klasa niejednoznaczności części mowy	(subst)

3.4.2. Kategorie morfologiczne

Dla wszystkich kategorii morfologicznych używane są następujące atrybuty:

Pozycja	Cecha	Przykładowa wartość dla przypadku
-2	część mowy	pred
-1	część mowy	qub
-1	słowo	jeszcze
0	słowo	jedna
0	ostatnia litera	a
0	ostatnie dwie litery	na
0	ostatnie trzy litery	dna
1	klasa niejednoznaczności kategorii	(nom,voc)

Dla poszczególnych kategorii morfologicznych używane są dodatkowe atrybuty wyszczególnione poniżej.

Liczba (ang. *number*)

Wartości: sg, pl.

Pozycja	Cecha
-2	liczba
-1	liczba
1	klasa niejednoznaczności liczby

Przypadek (ang. *case*)

Wartości: nom, gen, dat, acc, inst, loc, voc.

Pozycja	Cecha
-2	przypadek
-2	czy słowo równe „nie”?
-1	przypadek
-1	czy słowo zanegowane?
1	klasa niejednoznaczności przypadku

Rodzaj (ang. *gender*)

Wartości: m1, m2, m3, f, n.

Pozycja	Cecha
-2	rodzaj
-1	rodzaj
1	klasa niejednoznaczności rodzaju

Osoba (ang. *person*)

Wartości: pri, sec, ter (pierwsza, druga, trzecia).

Pozycja	Cecha
-2	osoba
-1	osoba
1	klasa niejednoznaczności osoby

Stopień (ang. *degree*)

Wartości: pos, com, sup (równy, wyższy i najwyższy).

Pozycja	Cecha
-2	stopień
-1	stopień
1	klasa niejednoznaczności stopnia

Aspekt (ang. *aspect*)

Wartości: perf, imprf (dokonany, niedokonany).

Pozycja	Cecha
0	pierwsza litera
0	pierwsze dwie litery
0	pierwsze trzy litery

Negacja (ang. *negation*)

Wartości: aff, neg.

Pozycja	Cecha
0	czy słowo zaczyna się na „nie”?

Akomodacyjność (ang. *accommodability*)

Wartości: congr, rec (uzgadniająca, rządząca).

Pozycja	Cecha
-2	przypadek
-1	przypadek
1	klasa niejednoznaczności przypadku

Interpunkcja (ang. *fullstoppedness*)

Wartości: pun, npun (segment powinien być zakończony kropką, brak kropki po segmencie).

Pozycja	Cecha
-2	przypadek
-1	przypadek
1	klasa niejednoznaczności przypadku

Pozostałe

Pozostałe kategorie morfologiczne:

- akcentowość (ang. *accentability*),
- poprzyimkowość (ang. *post-prepositionality*),
- aglutynacyjność (ang. *agglutination*),
- wokaliczność (ang. *vocalicity*),

korzystają ze standardowego zestawu schematów zdefiniowanego na początku podrozdziału 3.4.2.

Rozdział 4

Ewaluacja tagera

Porównanie skuteczności tagera maksimum entropii z innymi tagerami nie jest łatwe, nawet w obrębie tagerów dla języka polskiego. Korpusy różnią się między sobą zbiorem użytych tagów, niektóre wymagają dokładnie jednego poprawnego tagu przypisanego do słowa (nazywamy je korpusami w pełni ujednoznacznionymi, np. NKJP), a inne dopuszczają kilka prawidłowych wartości dla jednego słowa (np. Korpus Języka Polskiego IPI PAN). Tager stworzony dla korpusu w pełni ujednoznacznionego wybiera dokładnie jeden tag, natomiast dla korpusu dopuszczającego większą liczbę poprawnych wartości, przewiduje zbiór wszystkich poprawnych tagów. Nawet jeżeli rozpatrujemy tagery uczone i testowane na tym samym korpusie, oraz bazujące na tym samym zbiorze danych, to nie można założyć, że wyniki są w pełni porównywalne. Korpus ewoluuje, wprowadzane są korekty błędów, zmienia się zawartość korpusu poprzez dodawanie lub usuwanie tekstów.

Pojęcie skuteczności tagera jest bardzo nieprecyzyjne, z wielu prac naukowych nie wynika, w jaki sposób została ona policzona. W pracy [Karwańska i Przepiórkowski 2009] zostały opisane różne sposoby liczenia skuteczności, jak również została zaproponowana metodologia ewaluacji tagera. Opiera się ona na przeprowadzeniu 10-krotnej walidacji krzyżowej, w trakcie której liczone są następujące miary:

- **correctness** (procent słów, dla których tagi poprawne i wybrane przez tager zgadzają się w 100%),
- **weak correctness** (procent słów, dla których zbiory tagów poprawnych i wybranych przez tager mają niepuste przecięcie),
- **precision** (procent tagów wybranych przez tager, które należą do zbioru tagów poprawnych),
- **recall** (procent poprawnych tagów, które zostały wybrane przez tager),
- **F-measure** (średnia harmoniczna precision i recall, $F = \frac{2PR}{P+R}$).

Bardzo pomocny w zrozumieniu różnic między powyższymi miarami jest przykład zawarty w pracy [Karwańska i Przepiórkowski 2009].

Narodowy Korpus Języka Polskiego jest korpusem w pełni ujednoznacznionym, dla każdego słowa istnieje tylko jeden poprawny tag. Tager maksimum entropii wybiera dokładnie jeden najbardziej prawdopodobny tag dla słowa. Z punktu widzenia tagera MaxEnt wszystkie zaproponowane wyżej miary są sobie równe. W podrozdziale 4.4 został on jednak porównany do innych tagerów, których skuteczność była mierzona na korpusie IPI PAN. Dla tych tagerów podane wyżej miary nie są równe.

4.1. Metodologia i dane treningowe

Ewaluacji tagera dokonano poprzez 10-krotną walidację krzyżową przeprowadzoną na próbce korpusu wielkości około 1,2 mln segmentów. Teksty pochodzą z Narodowego Korpusu Języka Polskiego i zostały ręcznie otagowane niezależnie przez dwóch anotatorów. Konflikty tagowania były rozwiązywane przez superanotatora.

Uczenie następowało na około 9/10 wielkości korpusu, a testowanie na pozostałej 1/10 korpusu. Podczas kolejnych etapów walidacji krzyżowej wielkość korpusu treningowego wahała się między 1 029 000 a 1 120 000, a testowego od 95 000 do 186 000 segmentów. Poniżej przedstawiono średnie wielkości korpusów.

Opis	Wielkość
Średnia liczba segmentów w korpusie treningowym	1 096 049,8
Średnia liczba segmentów w korpusie testowym	121 783,2

Wszystkie modele dla klas niejednoznaczności części mowy AC_{POS} , które wystąpiły mniej niż 20 razy, oraz modele dla kategorii morfologicznych, które wystąpiły mniej niż 10 razy, zostawały odrzucane w procesie uczenia. Przykład: jeżeli klasa niejednoznaczności dla części mowy jest bardzo rzadka, np. $AC_{POS} = (adj, ger)$, to wybór części mowy następował na podstawie tagera unigramowego. Tager unigramowy wybierał część mowy lub wartość kategorii morfologicznej najczęściej występującą w korpusie treningowym razem z tagowanym słowem. Później przedstawiono średnie częstości użycia tagera unigramowego w procesie walidacji krzyżowej:

Kategoria	Średnia liczba wystąpień	Procentowa liczba wystąpień
część mowy	14	0,09 %
przypadek	1	0,02 %
akomodacyjność	0,3	0,02 %
stopień	0,8	0,01 %
wokaliczność	0,8	0,01 %
rodzaj	4,2	0,01 %
liczba	0,7	0,00 %
osoba	0,3	0,00 %
poprzyimkowość	0,2	0,01 %

4.2. Skuteczność modelu maksimum entropii

Średnia skuteczność tagera z 10-krotnej walidacji krzyżowej została przedstawiona w tabeli poniżej.

Kategoria	Średnia liczba prób tagowania	Średnia skuteczność
część mowy	121 783	98,31 %
cały tag	121 783	93,10 %
część mowy + przypadek	64 749	93,30 %
część mowy + akomodacyjność	1 613	97,52 %
część mowy + stopień	17 180	98,30 %
część mowy + wokaliczność	18 676	99,95 %
część mowy + interpunkcja	1 073	97,59 %
część mowy + rodzaj	58 359	93,14 %
część mowy + liczba	60 289	96,34 %
część mowy + negacja	8 382	91,90 %
część mowy + osoba	9 395	99,45 %
część mowy + poprzyimkowość	1 338	99,69 %
część mowy + aspekt	18 148	98,05 %
część mowy + akcentowość	2 148	99,51 %
część mowy + aglutynacyjność	5 341	100,00 %

Należy pamiętać, że często klasa niejednoznaczności posiada tylko jedną wartość. Wtedy wybór jest jednoznaczny i tager nie ma możliwości popełnienia błędu. Poniżej przedstawiono procentowy udział jednoelementowych klas niejednoznaczności we wszystkich próbach tagowania danej kategorii. Próba tagowania kategorii morfologicznej następowała tylko przy prawidłowo wybranej części mowy.

Kategoria	Średnia liczba wystąpień	Częstość występowania jednej wartości
część mowy	83 769	68,72 %
przypadek	19 891	30,67 %
akomodacyjność	1 338	82,87 %
stopień	16 799	97,78 %
wokaliczność	18 666	99,95 %
interpunkcja	791	74,68 %
rodzaj	37 819	64,68 %
liczba	48 836	82,00 %
negacja	7 073	90,30 %
osoba	9 340	99,42 %
poprzyimkowość	1 021	76,25 %
aspekt	17 371	95,71 %
akcentowość	1 978	92,04 %
aglutynacyjność	5 341	100,00 %

W wynikach najbardziej dziwią niskie, biorąc pod uwagę częstość występowania jednej wartości, skuteczności tagowania kategorii osoby i negacji. Wydaje się, że zastosowanie tutaj innej metody wyboru poprawnej wartości dałoby lepszą skuteczność. Wybór częściej występującej z dwóch dopuszczalnych wartości dla negacji powinien dać skuteczność minimum 50% w przypadku dezambiguacji dwóch wartości.

4.3. Tager MaxEnt dla języka czeskiego

Motywacją do zbudowania tagera maksimum entropii dla języka polskiego były bardzo dobre wyniki uzyskane dla języka czeskiego w pracy [Hajič i Hladká 1998]. Stamtąd został zaczerpnięty pomysł zastosowania modelu do klas niejednoznaczności, a także pomysł tagowania oddzielnie części mowy i kategorii morfologicznych. Wyniki uzyskane przez tager dla języka czeskiego są podane w poniższej tabeli.

Tager	Język	Skuteczność
MaxEnt	PL	93,10 %
MaxEnt	CZ	92,38 %
MaxEnt VTC	CZ	93,78 %

Różnice między tagerami dla języka polskiego i dla języka czeskiego:

- Hajič i Hladká dezambiguują tylko tagi, ignorują różne wartości form podstawowych. MaxEnt PL wybiera poprawną formę podstawową słowa.
- MaxEnt dla kategorii AC_{POS} tworzy oddzielne modele dla różnych części mowy. Z pracy [Hajič i Hladká 1998] nie jest do końca jasne, jak zostało to rozwiązane dla MaxEnt CZ. Wydaje się, że został zbudowany tylko jeden model dla AC_{POS} .
- MaxEnt PL stosuje prawdopodobieństwa unigramowe jako backup (tzn. wtedy, kiedy nie mamy modelu MaxEnt dla napotkanej klasy niejednoznaczności). Hajič i Hladká użyli wygładzania danego wzorem:

$$p_{AC}(t|k) = \sigma p_{AC,e}(t|k) + (1 - \sigma)p_{AC,1}(t), \quad (4.1)$$

gdzie $p_{AC,e}$ jest prawdopodobieństwem z modelu MaxEnt, a $p_{AC,1}$ prawdopodobieństwem z modelu unigramowego.

- Tager MaxEnt CZ używa bardziej złożonych cech kontekstowych (zobacz podrozdział 4.6) oraz inną definicję wartości cechy kontekstowej:

$$f_{AC_{Cat}, \bar{k}, \bar{t}}(k, t) = 1 \Leftrightarrow \bar{t} = t \wedge \bar{k} \subset k, \quad (4.2)$$

gdzie AC_{Cat} to klasa niejednoznaczności AC kategorii morfologicznej Cat , a (\bar{k}, \bar{t}) jest wartością kontekstu i tagu, przy użyciu których cecha została zdefiniowana. W tagerze MaxEnt PL jest:

$$f_{Cat_{AC}, \bar{k}, \bar{t}}(k, t) = 1 \Leftrightarrow \bar{t} = t \wedge \bar{k} = k. \quad (4.3)$$

Zauważmy, że przy cechach prostych użytych w MaxEnt PL, obie definicje są równoważne.

- Parametry λ_i modelu MaxEnt PL zostały wyuczone metodą Newtona, MaxEnt CZ używa tylko estymatorów (prostszych i mniej dokładnych, ale szybszych).
- W tagerze dla języka czeskiego MaxEnt VTC została użyta metoda VTC (ang. *Valid subTag Combinations*). Zauważmy, że w przypadku niezależnego wyboru k kategorii morfologicznych dla tagu może się zdarzyć, że wybrane wartości c_1, c_2, \dots, c_n kategorii morfologicznych C_1, C_2, \dots, C_n (odpowiednio) nie tworzą dopuszczalnego tagu (czyli tagu należącego do zbioru możliwych tagów wybranych przez Morfeusza). Hajič i Hladká zastosowali metodę VTC, która:

- niezależnie dla każdej kategorii C_j liczy prawdopodobieństwa wystąpienia wartości z klasy niejednoznaczności tej kategorii;
- liczy prawdopodobieństwa dopuszczalnych tagów $t^i = (c_1^i, c_2^i, \dots, c_n^i)$ jako iloczyn prawdopodobieństw jego wartości $c_1^i, c_2^i, \dots, c_n^i$ w poszczególnych kategoriach, tj:

$$p(t^i) = p(c_1^i)p(c_2^i) \dots p(c_n^i); \quad (4.4)$$

- wybierana tag o największej wartości $p(t^i)$.

Takie postępowanie pozwala na podwyższenie skuteczności tagera i wyeliminowanie przypadków, w których najbardziej prawdopodobne wartości kategorii morfologicznych nie tworzą dopuszczalnego tagu. W modelu MaxEnt PL metoda VTC jeszcze nie została zastosowana. Jej implementacja na pewno przyniesie poprawę skuteczności tagera.

4.4. Porównanie skuteczności tagerów dla języka polskiego

Do porównania skuteczności wybrano następujące tagery:

- Tager **Trigram HMM** (ang. *Hidden Markov Model*) Łukasza Dębowskiego oparty na trigramowym ukrytym modelu Markowa. Został opisany w pracy [Dębowski 2004].
- **TaKIPI** oparty na regułach decyzyjnych, unigramowych rozkładach prawdopodobieństwa i ręcznych regułach poprawiających błędy. Tager TaKIPI został opisany w pracy [Piasecki i Godlewski 2006].
- Tager **Brilla** autorstwa Szymona Acedańskiego oparty na transformacjach. Został on, w stosunku do pierwowzoru, znacznie przyspieszony i dostosowany do języków fleksyjnych. Więcej szczegółów można znaleźć w pracy [Acedański 2010].

Tagery Trigram HMM, TaKIPI oraz tager Brilla były testowane na korpusie IPI PAN. Wyniki ich ewaluacji (w procentach) znajdują się w tabeli poniżej. Źródła: [Acedański 2010], [Karwańska i Przepiórkowski 2009].

Tager	Cały tag					Część mowy				
	C	WC	P	R	F	C	WC	P	R	F
Trigram HMM	87,39	90,59	84,51	83,09	83,80	96,79	97,11	96,75	96,78	96,77
TaKIPI	88,68	91,06	90,94	83,78	87,21	96,53	96,54	96,58	96,71	96,65
Brill	90,00	92,44	92,44	86,05	89,13	98,17	98,18	98,18	98,16	98,17

Powyższe skróty oznaczają następujące miary (zdefiniowane we wstępie do tego rozdziału):

- **C** = Correctness,
- **WC** = Weak Correctness,
- **P** = Precision,
- **R** = Recall,
- **F** = F-measure.

Tager Brilla był również testowany na korpusie NKJP, podobnie jak omawiany w tej pracy tager maksimum entropii. Skuteczność w tym przypadku oznacza każdą z miar $C = WC = P = R = F$.

Tager	Skuteczność
Brill	92,82 %
MaxEnt	93,10 %

Tagera Brilla na korpusie IPI PAN wypada najlepiej spośród wybranych tagerów. Widzimy też, że skuteczność tagera Brilla jest wyższa dla korpusu NKJP (92,82%) niż dla korpusu IPI PAN (92,44%). Tager maksimum entropii z wynikiem 93,1% na korpusie NKJP wypada najlepiej na tle konkurentów. Trzeba jednak zaznaczyć, że został wytrenowany na największym i najbardziej dopracowanym zbiorze danych.

4.5. Analiza błędów

W tym podrozdziale przedstawiono skuteczność tagera w dezambiguacji poszczególnych części mowy. W czwartej kolumnie znajduje się najczęściej wybierana błędnie wartość przez tager, a w kolumnie piątej procent wyboru tej wartości. Pusta wartość w kolumnie *Najczęstszy błąd* oznacza, że model unigramowy (użyty jako backup) przypisywał pustą wartość, czyli tagowane słowo nie wystąpiło w tekście treningowym i model unigramowy nie był w stanie przypisać żadnego sensownego tagu dla tego słowa. Wszystkie wartości zostały uśrednione z walidacji krzyżowej i zaprezentowane w tabeli 4.1.

Najwięcej błędów popełnianych jest w najtrudniejszych do dezambiguacji kategoriach przypadku i rodzaju. Podobne trudności z tymi dwoma kategoriami mają inne tagery, więc nie jest to zaskoczeniem. Tager w żaden sposób nie używa ręcznie przygotowanych reguł gramatycznych. Wszystkie błędy mają charakter statystyczny. Z tego powodu ciężko zauważyć jakąkolwiek zależność między błędami jednego rodzaju (czyli np. przypisanie mianownika zamiast biernika). Tym bardziej trudno dobrać cechą kontekstową, która rozwiązywałaby dany problem.

Tabela 4.1: Najczęstsze błędy dla części mowy

POS	Liczba wystąpień	Skuteczność	Najczęstszy błąd	Częstość błędu
interp	22 416	100,00%		0,00%
aglt	760	100,00%		0,00%
inf	1 925	99,91%	subst	0,04%
siebie	213	99,89%		0,07%
praet	5 341	99,80%	adj	0,10%
ppron3	1 338	99,80%	ppron12	0,06%
bedzie	280	99,70%		0,27%
pcon	262	99,64%		0,36%
ppron12	811	99,49%	ppron3	0,34%
fin	5 958	99,43%	subst	0,28%
imps	211	99,42%		0,38%
prep	11 599	99,29%	conj	0,22%
adja	60	99,17%	adjp	0,54%
Dokończenie na następnej stronie				

Tabela 4.1: Najczęstsze błędy dla części mowy (dok.)

POS	Liczba wystąpień	Skuteczność	Najczęstszy błąd	Częstość błędu
pant	15	98,95%	prep	0,71%
winien	81	98,56%	adj	0,95%
adjp	79	98,54%	adj	0,53%
subst	33 208	98,39%	adj	0,56%
adj	12 889	98,22%	subst	0,86%
brev	1 074	98,09%	prep	1,14%
comp	1 789	97,84%	qub	0,95%
num	1 600	97,80%	adv	1,62%
adv	4 292	97,06%	qub	1,23%
impt	248	96,46%	subst	2,16%
conj	4 491	95,99%	qub	2,07%
pact	551	95,47%	adj	3,56%
qub	6 828	95,27%	conj	1,63%
ppas	1 342	94,49%	adj	4,60%
numcol	13	91,89%	num	3,66%
adjc	9	87,82%		4,50%
ger	1 174	85,71%	subst	14,05%
interj	156	87,34%	prep	4,66%
xxx	89	84,53%	subst	4,04%
pred	656	84,12%	subst	8,11%
burk	14	81,66%	subst	10,63%
depr	12	64,73%	subst	22,19%

Poniżej znajdują się tablice kontyngencji ilustrujące błędy popełniane przez tager. Jeden wiersz odpowiada poprawnej wartości kategorii, a kolumna odpowiada wartości wybranej przez tager. Na przecięciu i -tego wiersza i j -tej kolumny mamy procent przypadków, w których tager przypisał wartość j , a prawidłowa wartość to i . Pusta wartość procentowa oznacza 0,00%. Ponadto poprawna pusta wartość oznacza, że atrybut był opcjonalny i nie był podany w poprawnym tagu. W drugiej kolumnie znajdują się średnie liczby wystąpień w korpusie testowym z procesu walidacji krzyżowej.

Przypadek

	liczba	nom	gen	dat	acc	inst	loc	voc	
nom	15 643	91,38%	1,37%	0,01%	4,10%	0,01%	0,03%	0,02%	3,08%
gen	19 855	0,60%	96,76%	0,11%	0,80%	0,11%	0,25%		1,38%
dat	1 401	0,75%	5,00%	88,71%	0,39%	1,08%	1,60%	0,01%	2,46%
acc	12 310	8,36%	3,11%	0,03%	85,56%	0,20%	0,99%	0,00%	1,76%
inst	5 000	0,10%	0,37%	0,09%	0,79%	96,99%	0,18%		1,48%
loc	10 368	0,06%	0,85%	0,08%	0,39%	0,12%	97,72%	0,02%	0,78%
voc	172	8,82%	0,77%	0,18%	0,89%		4,49%	82,28%	2,58%

Rodzaj

	liczba	m1	m2	m3	f	n	
m1	11 817	93,58%	0,29%	2,76%	1,16%	0,54%	1,67%
m2	793	12,90%	69,10%	9,45%	3,40%	3,16%	1,99%
m3	16 165	2,74%	0,17%	93,75%	1,28%	1,31%	0,75%
f	18 717	1,42%	0,05%	1,14%	95,63%	0,90%	0,86%
n	10 867	0,81%	0,06%	2,64%	1,80%	89,22%	5,46%

Osoba

	liczba	pri	sec	ter	
pri	2 501	99,84%			0,16%
sec	833		98,51%		1,49%
ter	6 061	0,01%		99,43%	0,56%

Stopień

	liczba	pos	com	sup	
pos	14 653	98,09%			1,91%
com	5 112	0,02%	97,96%		2,02%
sup	278			99,47%	0,54%
	1 738	0,01%			99,99%

4.6. Pomysły na usprawnienie tagera

Największy wzrost skuteczności tagera mogą przynieść eksperymenty z różnymi konfiguracjami cech kontekstowych. Dobór odpowiednich cech, oddzielnie dla każdej kategorii morfologicznej, jest w modelu maksimum entropii kluczowy. Wydaje się również, że można zastosować bardziej złożone cechy kontekstowe niż zaproponowane w tej pracy. Przykładem bardziej złożonej cechy kontekstowej może być:

$$f(k, t) = \begin{cases} 1 & \text{jeżeli } t = \text{subst i } k(\text{tag}, -1) = \text{adj i } k(\text{słowo}, 0) = \text{„stół”} \\ 0 & \text{wpp.} \end{cases} \quad (4.5)$$

W przedstawionym tagerze zostały użyte tylko cechy z prostszymi warunkami, zawierającymi tylko jeden atrybut z kontekstu i wartość tagu:

$$f(k, t) = \begin{cases} 1 & \text{jeżeli } t = \text{subst i } k(\text{słowo}, 0) = \text{„stół”} \\ 0 & \text{wpp.} \end{cases} \quad (4.6)$$

Bardzo ważnym usprawnieniem może być również zastosowanie metody VTC użytej w tagerze MaxEnt dla języka czeskiego. Jej krótki opis znajduje się w podrozdziale 4.3.

Świetne wyniki może dać połączenie tagera statystycznego MaxEnt z innym tagerem T (np. tagerem opartym na regułach lingwistycznych, tagerem Brilla opartym na transformacjach lub tagerem opartym na drzewach decyzyjnych). Możemy tutaj wyróżnić 3 możliwe podejścia:

- Zawężenie klasy niejednoznaczności przez tager T, następnie dezambiguacja tagerem MaxEnt.
- Dezambiguacja tagerem MaxEnt, następnie poprawianie błędów tagerem T.

- Równoczesne używanie tagerów T i MaxEnt, tzn. równoległe tagowania dwoma tagerami i wybór tagu o największej sumie prawdopodobieństw w obu modelach.

Rozdział 5

Implementacja

Implementacja została wykonana w języku Python, korzysta z pakietu *nlTK.classify.maxent* i umożliwia rozproszone budowanie modelu. Skrypty uruchamiające zostały napisane w języku Bash. Uruchamianie procesów na zdalnych maszynach odbywa się poprzez protokół SSH. W aplikacji został zawarty mechanizm, który uodparnia aplikację na awarie zdalnych maszyn. Jeżeli zadanie przydzielone zdalnej maszynie nie zakończy się sukcesem, to jest przekazywane innej dostępnej maszynie. Każda maszyna trenuje inny podzbiór wszystkich kilkuset modeli MaxEnt. Wyniki otrzymane od zdalnych maszyn są zbierane w centralnym systemie plików, wspólnym dla wszystkich maszyn.

Działanie aplikacji możemy podzielić na 3 fazy:

1. Wczytywanie danych treningowych z formatu XML, stworzenie wewnętrznych struktur danych i zapis do plików (na maszynie głównej).
2. Rozproszone budowanie około 350 modeli MaxEnt (na maszynie głównej i maszynach zdalnych). Maszyny zdalne wczytują struktury danych z plików, trenują przydzielony podzbiór modeli, a następnie zapisują wyniki do plików.
3. Testowanie na korpusie testowym (wykonywane na głównej maszynie). Maszyna wczytuje wytrenowane modele i stosuje je do danych testowych.

Pierwsza i trzecia faza razem zajmują około 2 minut dla jednego etapu walidacji krzyżowej. Długość fazy drugiej w znaczącym stopniu zależy od górnego ograniczenia na liczbę iteracji wykonywanych w metodzie Newtona uczenia modelu. Zwiększenie liczby iteracji ponad 150 praktycznie nie wpływa na poprawę skuteczności tagera, natomiast znacznie wydłuża czas budowania modeli.

Obliczenia zostały wykonane w laboratorium Wydziału Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. Do obliczeń wykorzystano 60 maszyn o następujących parametrach: procesor Intel Core 2 Duo 2,4–3,0 GHz, pamięć RAM 2–4 GB. Na próbie korpusu o wielkości 1,2 mln segmentów wykonano dziesięciokrotną walidację krzyżową (uczenie na 9/10 korpusu, testowanie na pozostałej 1/10). Obliczenia zostały wykonane przy ograniczeniu równym 150 na liczbę iteracji w metodzie Newtona uczenia modelu. Wykonanie dziesięciokrotnej walidacji krzyżowej na 60 maszynach trwa około 14 godzin. Obciążenie maszyn podczas obliczeń nie jest równomierne. Szybkość działania nie była priorytetem przy implementacji. Wydajność tagera na pewno można zwiększyć w znaczny sposób, implementując model maksimum entropii w bardziej wydajnym języku programowania oraz dostosowując go do specyfiki korpusu NKJP.

Opis uruchamiania skryptów do walidacji krzyżowej znajduje się w dodatku B.1 oraz w pliku *readme* na płycie. Na płycie znajdują się też skrypty do przeglądania najczęstszych

błędów popełnianych przez tager. Jest tam również przykładowy plik z korpusu NKJP użyty do ewaluacji tagera. Cały korpus nie jest jeszcze oficjalnie dostępny. Zawartość płyty CD dołączonej do pracy została opisana w dodatku C.

Zakończenie

Moja praca magisterska, o ile mi wiadomo, jest pierwszą próbą zastosowania modelu maksimum entropii do tagowania języka polskiego. Uzyskaną skuteczność tagera na poziomie 93,1% można uznać za sukces. Tak wysoki wynik pokazuje ogromny potencjał drzemiący w tym modelu. Skuteczność tagera MaxEnt na pewno można w znaczącym stopniu poprawić. Może to być bardzo dobry temat na kolejną pracę magisterską.

Serdecznie dziękuję mojemu promotorowi dr. hab. Adamowi Przepiórkowskiemu, któremu zawdzięczam pomysł na tą pracę magisterską, a także cenne uwagi i wskazówki lingwistyczne dotyczące wyboru cech kontekstowych dla modeli maksimum entropii.

Dodatek A

Metoda mnożników Lagrange’a

A.1. Intuicje geometryczne

Metoda mnożników Lagrange’a jest metodą używaną w optymalizacji. Określa ona strategię znajdowania minimum lub maksimum funkcji w zbiorze wyznaczonym przez zbiór równań. Weźmy problem maksymalizacji funkcji $f : X \rightarrow \mathbb{R}$, $X \subset \mathbb{R}^n$, na zbiorze określonym warunkami $g(x) = 0$, gdzie $g : X \rightarrow \mathbb{R}^m$. Przyjmijmy, że $f, g \in \mathcal{C}^1$ (są różniczkowalne i mają ciągłą pochodną) oraz $\mathcal{G} = \{x \in X : g(x) = 0\}$. Problem maksymalizacji funkcji f na zbiorze \mathcal{G} określamy równaniem:

$$\max_{\substack{x \in X \\ g(x)=0}} f(x). \quad (\text{A.1})$$

Przypuśćmy, że funkcja f na zbiorze \mathcal{G} przyjmuje maksimum w punkcie x^* . Wtedy x^* musi spełniać następujące warunki:

- $x^* \in \mathcal{G}$, tzn.

$$\begin{aligned} g_1(x^*) &= 0, \\ g_2(x^*) &= 0, \\ &\vdots \\ g_m(x^*) &= 0. \end{aligned} \quad (\text{A.2})$$

- Poruszając się po zbiorze \mathcal{G} w otoczeniu x^* , funkcja f nie rośnie. Po zbiorze \mathcal{G} możemy się poruszać tylko w tych kierunkach, w których żadna z funkcji g_i nie zmienia wartości, aby warunki (A.2) były dalej spełnione. Funkcja g_i wzrasta w kierunku ∇g_i (gradient g_i), więc ten kierunek jest zakazany dla $i \in \{1, 2, \dots, m\}$. Zbiór zakazanych kierunków tworzy następującą przestrzeń liniową:

$$\text{lin}\{\nabla g_1, \nabla g_2, \dots, \nabla g_m\} = \left\{ v \in \mathbb{R}^m \mid \exists \lambda_1, \dots, \lambda_m \in \mathbb{R} \quad v = \sum_{i=1}^m \lambda_i \nabla g_i \right\}. \quad (\text{A.3})$$

Chcemy, żeby nie dało się zwiększyć wartości $f(x)$, poruszając się po zbiorze \mathcal{G} w otoczeniu x^* , więc ∇f musi należeć do zbioru zakazanych kierunków. Dzieje się tak wtedy, gdy:

$$\exists \lambda_1, \dots, \lambda_m \in \mathbb{R} \quad \nabla f(x^*) = \sum_{i=1}^m -\lambda_i \nabla g_i(x^*) \quad (\text{A.4})$$

W równaniu (A.4) użyliśmy mnożników $-\lambda_i$ zamiast λ_i z uwagi na elegancję dalszych rozważań. Nie ma to znaczenia, ponieważ możemy wybrać dowolne $\lambda \in \mathbb{R}^m$. Zapiszmy teraz dokładniej warunki (A.2) i (A.4) w postaci $n + m$ równań skalarnych:

$$\begin{aligned}
g_1(x^*) &= 0, \\
g_2(x^*) &= 0, \\
&\vdots \\
g_n(x^*) &= 0, \\
\frac{\partial f}{\partial x_1}(x^*) + \sum_{i=1}^m \lambda_i \frac{\partial g_i}{\partial x_1}(x^*) &= 0, \\
\frac{\partial f}{\partial x_2}(x^*) + \sum_{i=1}^m \lambda_i \frac{\partial g_i}{\partial x_2}(x^*) &= 0, \\
&\vdots \\
\frac{\partial f}{\partial x_n}(x^*) + \sum_{i=1}^m \lambda_i \frac{\partial g_i}{\partial x_n}(x^*) &= 0.
\end{aligned} \tag{A.5}$$

Zauważmy, że poszukiwanie punktów x^* spełniających warunki (A.5) dla pewnego λ jest tym samym, co poszukiwanie punktów stacjonarnych (czyli takich, w których zeruje się pochodna) funkcji $\Lambda : X \times \mathbb{R}^m \rightarrow \mathbb{R}$ określonej wzorem:

$$\Lambda(x, \lambda) \equiv f(x) + \sum_i \lambda_i g_i(x). \tag{A.6}$$

Żeby to sprawdzić, wystarczy policzyć pochodne cząstkowe funkcji $\Lambda(x, \lambda)$:

$$\begin{aligned}
\frac{\partial \Lambda}{\partial x_j}(x) &= \frac{\partial f}{\partial x_j}(x) + \sum_{i=1}^m \lambda_i \frac{\partial g_i}{\partial x_j}(x), \\
\frac{\partial \Lambda}{\partial \lambda_j}(x) &= g_j(x),
\end{aligned} \tag{A.7}$$

a następnie zauważyć, że przyrównując je do 0 otrzymujemy dokładnie warunki (A.5).

Zauważmy, że funkcja Λ może mieć więcej punktów stacjonarnych. Który argument w takim przypadku powinniśmy wybrać za $x^* = \arg \max_{x \in \mathcal{G}} f(x)$? Niech dla ustalonego λ :

$$x_\lambda = \arg \max_x \Lambda(x, \lambda). \tag{A.8}$$

Wtedy zachodzi nierówność:

$$\Lambda(x_\lambda, \lambda) \geq \Lambda(x^*, \lambda). \tag{A.9}$$

Zauważmy teraz, że $\Lambda(x^*, \lambda)$ nie zależy od λ , ponieważ:

$$\Lambda(x^*, \lambda) = f(x^*) + \sum_i \lambda_i g_i(x^*) = f(x^*). \tag{A.10}$$

Możemy zatem napisać:

$$\min_\lambda \Lambda(x_\lambda, \lambda) \geq f(x^*) \tag{A.11}$$

Jeżeli funkcje f i g spełniają dodatkowe warunki:

- funkcja $f(x)$ jest wklęsła (tzn. każda z funkcji $f_i(x)$ jest wklęsła),
- funkcja $g(x)$ jest funkcją liniową,

to prawdziwa jest równość:

$$\min_{\lambda} \Lambda(x_{\lambda}, \lambda) = f(x^*). \quad (\text{A.12})$$

W niniejszej pracy są rozpatrywane tylko te problemy, dla których spełnione są powyższe warunki. Równanie (A.12) możemy zapisać w następującej postaci:

$$\min_{\lambda} \max_x \Lambda(x, \lambda) = f(x^*). \quad (\text{A.13})$$

A.2. Algorytm

Formalnie algorytm rozwiązywania problemu określonego w (A.1) składa się z następujących kroków.

1. Dla każdego warunku $g_i(x) = 0$ wprowadzamy dodatkową zmienną λ_i oraz określamy funkcję $\Lambda : X \times \mathbb{R}^n \rightarrow \mathbb{R}$:

$$\Lambda(x, \lambda) = f(x) + \sum_{i=1}^m \lambda_i g_i(x). \quad (\text{A.14})$$

2. Spośród zbioru punktów stacjonarnych funkcji Λ wybieramy x^* spełniające równanie (A.13). Zwykle ten krok podzielony jest na dwa mniejsze:

- (a) Ustalamy λ i traktujemy funkcję $\Lambda(x, \lambda)$ jako funkcję od x . Dla każdego λ znajdujemy punkt stacjonarny x_{λ} spełniający warunek:

$$x_{\lambda} = \arg \max_{x \in X} \Lambda(x, \lambda). \quad (\text{A.15})$$

- (b) Traktujemy $\Lambda(x_{\lambda}, \lambda)$ jako funkcję od λ . Wprowadzamy funkcję $\Psi(\lambda) \equiv \Lambda(x_{\lambda}, \lambda)$, którą nazywamy funkcją dualną do f . Szukamy punktów stacjonarnych funkcji Ψ , następnie spośród nich wybieramy argument λ^* , dla którego Ψ przyjmuje najmniejszą wartość:

$$\lambda^* = \arg \min_{\lambda} \Psi(\lambda), \quad (\text{A.16})$$

$$x^* = x_{\lambda^*}. \quad (\text{A.17})$$

3. Wnioskujemy, że:

$$x^* = \arg \max_{\substack{x \in X \\ g(x)=0}} f(x). \quad (\text{A.18})$$

Zwykle krok (2a) jest łatwiejszy niż (2b), tzn. wzór na x_{λ} w zależności od λ udaje się wyprowadzić w sposób analityczny. Krok (2b) bywa trudniejszy i często nie udaje się w sposób analityczny znaleźć λ^* . Stosuje się wtedy różnego rodzaju metody przybliżeń, siecznych lub stycznych, które wywodzą się z wielowymiarowej metody Newtona. Dowód wniosku (A.18) zamieszczamy poniżej.

Twierdzenie A.2.1 (Lagrange). *Ustalmy λ^* . Przypuśćmy, że istnieje $x^* \in X$ takie, że $x^* = \arg \max_{x \in X} \Lambda(x, \lambda^*)$ oraz $g(x^*) = 0$. Wtedy argument x^* maksymalizuje $f(x)$ na zbiorze X .*

Dowód.

$$\begin{aligned}\max_{\substack{x \in X \\ g(x)=0}} f(x) &= \max_{\substack{x \in X \\ g(x)=0}} [f(x) + \sum_{i=1}^n \lambda_i^* g_i(x)] \\ &\leq \max_{x \in X} [f(x) + \sum_{i=1}^n \lambda_i^* g_i(x)] \\ &= f(x^*) + \sum_{i=1}^n \lambda_i^* g_i(x^*) \\ &= f(x^*).\end{aligned}\tag{A.19}$$

□

Dodatek B

Dokumentacja użytkowa programu

B.1. Instalacja

Wszystkie stworzone w ramach tej pracy magisterskiej programy przeznaczone są dla systemu Linux. Do uruchomienia wymagają zainstalowanego środowiska Python 2.6.5 wraz z pakietem NLTK (Natural Language Toolkit). Wszystkie programy i skrypty udostępniane są na licencji GPL 3.0. Szczegóły są zawarte w pliku gpl-3.0.txt.

Przygotowanie do uruchomienia aplikacji powinno składać się z następujących kroków:

1. Skopiowanie katalogu z płyty na maszynę (np. w laboratorium na wydziale MIM UW). Do tego samego katalogu docelowego należy skopiować katalog z korpusem <korpus>. Na płycie znajduje się tylko jeden przykładowy plik ann_morphosyntax.xml z korpusu NKJP. Cały korpus nie jest jeszcze oficjalnie dostępny. Udostępnienie pełnej wersji korpusu przewidziane jest na 12 czerwca 2011 roku.
2. Stworzenie korpusu do 10-krotnej walidacji krzyżowej poleceniem:

```
./createCorpusSets.py <korpus>
```

Wynikiem działania programu będzie katalog <korpus>_subsets, w którym będzie znajdowało się 10 rozłącznych podzbiorów korpusu.

3. Edycja pliku corpusInfo.py w celu podania na zmienną corpusRoot nazwy katalogu, w którym znajdują się teksty do 10-krotnej walidacji krzyżowej, czyli <korpus>_subsets.
4. Edycja pliku machinesAll. Tam powinny znaleźć się nazwy wszystkich maszyn, których chcemy użyć w obliczeniach rozproszonych. Jeżeli aplikacja będzie uruchamiana w laboratorium MIM UW, to można pozostawić go bez zmian.
5. Za pomocą skryptu:

```
./chooseOkMachines.sh > machines
```

spośród maszyn z pliku machinesAll wybieramy te komputery, które są aktualnie dostępne (odpowiadają na żądania połączenia SSH).

6. Rozpoczynamy walidację krzyżową poleceniem:

```
./runDistributed.sh > <wyniki>
```

Statystyki działania tagera z kolejnych kroków walidacji krzyżowej zostaną zapisane w pliku <wyniki>.

7. W trakcie działania programu możemy monitorować zadania wykonywane na maszynach zdalnych poleceniem:

```
./ps.sh
```

Wyniki zapisywane są w pliku psRes.txt.

8. Jeżeli użytkownik chce zakończyć zadania wykonywane na maszynach zdalnych, to powinien wykonać komendę:

```
./kill.sh
```

9. Po zakończeniu działania programu można oglądać błędy popełniane przez tager, wykonując polecenie:

```
./showErrors <subset> <attrName> <correctVal> <predVal>
```

gdzie:

- <subset> to liczba naturalna z przedziału od 0 do 9 oznaczająca numer kroku walidacji krzyżowej,
- <attrName> to nazwa kategorii, której błędy chcemy oglądać np. ctag, case, gender,
- <correctVal> to poprawna wartość dla tej kategorii, np. nom,
- <predVal> to wartość przewidziana przez tager, np. acc.

Wykonanie polecenia:

```
./showErrors 2 case nom acc
```

spowoduje wyświetlenie wszystkich błędów wykonanych na zbiorze testowym nr 2 dla przypadku, gdzie tager zamiast mianownika wybrał biernik.

B.2. Opis poszczególnych programów i zawartości plików

W tej części znajdują się krótkie opisy plików zawartych na płycie CD. Opis działania programu zawsze dotyczy jednego etapu walidacji krzyżowej.

corpusInfo.py

W tym pliku zapisana jest ścieżka do korpusu. Jest on używany przez wszystkie pozostałe programy, oprócz createCorpusSets.py.

createCorpusSets.py

Program do dzielenia korpusu na 10 mniej więcej równych części. Uruchamianie komendą:

```
./createCorpusSets.py <korpus>
```

Wynikiem działania programu będzie katalog <korpus>_subsets, w którym będzie znajdowało się 10 rozłącznych podzbiorów korpusu.

maxEnt.py

Jest to klasa zawierająca implementację tagera. Nie uruchamia się jej bezpośrednio. Jeżeli użytkownik jest zainteresowany rozwijaniem aplikacji, na płycie może znaleźć plik `example.py` ilustrujący użycie klasy `maxEnt` do tagowania.

preDistributed.py

Jest to program wykonywany przez maszynę główną w pierwszej fazie obliczeń. Wykonywane są następujące kroki:

1. Parsowanie korpusu treningowego.
2. Stworzenie struktur danych potrzebnych do uczenia modeli.
3. Zapis stworzonych struktur danych do plików; pliki są umieszczane w podkatalogu `_model` utworzonego w katalogu zawierającym korpus testowy dla aktualnego etapu walidacji krzyżowej (czyli np. `<korpus>/subset2`).

learnSelectedModels.py

Ten program uruchamiany jest na maszynach zdalnych. Wykonuje on następujące operacje:

1. Wczytanie struktur danych z katalogu `_model`.
2. Trenowanie wybranych dla tej maszyny modeli (modele wybierane na podstawie podanego parametru).
3. Wyuczone modele dla części mowy zapisywane są w katalogu `_tagMaxentModels`, modele dla kategorii morfologicznych w katalogu `_msdMaxentModels`.
4. Jako podkatalogi katalogu zawierającego korpus testowy tworzone są dwa katalogi: `_tag_most_informative_features` oraz `_msd_most_informative_features`. Są tam zapisywane parametry λ_i wyuczone w metodzie Newtona. Plik `subst_acc_nom` w katalogu `_msd_most_informative_features/case` oznacza, że model został zbudowany dla rzeczownika i służy do dezambiguacji klasy niejednoznaczności przypadku $AC_{CASE} = (acc, nom)$.

postDistributed.py

Program uruchamiany jest na maszynie głównej po zakończeniu obliczeń rozproszonych. Wykonuje on następujące zadania:

1. Wczytanie modeli MaxEnt z katalogów `_tagMaxentModels` oraz `_msdMaxentModels`.
2. Parsowanie korpusu testowego.
3. Tagowanie w danych kontekstach.
4. Zapis wyników.

example.py

Przykładowy program pokazujący w jaki sposób należy korzystać z klasy `maxEnt` zawartej w pliku `maxEnt.py`.

skrypty Bash

- **chooseOkMachines.sh** Skrypt wybierające spośród maszyn zapisanych w pliku `machinesAll` te, które są aktualnie dostępne (odpowiadają na żądania połączenia ssh).
- **kill.sh** Skrypt pozwala zakończyć wszystkie zadania aktualnie wykonywane na zdalnych maszynach.
- **ps.sh** Skrypt pozwala na monitorowanie aktualnego obciążenia maszyn w trakcie obliczeń rozproszonych. W pliku `psRes.txt` zapisywane są wyniki działania programu `ps aux` na maszynach zdalnych.
- **runDistributed.sh** Jest to główny skrypt uruchamiający proces 10-krotnej walidacji krzyżowej. Argumentem tego programu jest zbiór 10 części korpusu. Argument podawany jest poprzez ustawienie zmiennej w pliku `corpusInfo.py`.

pozostałe pliki

- **gpl-3.0.txt** Treść licencji GNU GENERAL PUBLIC LICENSE, na której udostępniany jest program.
- **machines**
Zawiera spis wszystkich aktualnie odpowiadających maszyn spośród maszyn zapisanych w pliku `machinesAll`. Jest on uaktualniany ręcznie przez użytkownika poleceniem:

```
./chooseOkMachines > machines
```

- **machinesAll**
Zawiera spis wszystkich maszyn, których chcielibyśmy użyć do obliczeń rozproszonych.
- **readme**
Zawiera informacje zawarte w tym dodatku.
- **results.txt**
Jest to przykładowy wynik działania skryptu `runDistributed.sh`. Zawiera statystyki dotyczące walidacji krzyżowej.
- **results.ods**
Zawiera uśrednione wyniki walidacji krzyżowej obliczone na podstawie wyników z pliku `results.txt`.
- **korpus/text.xml**
Przykładowy tekst z NKJP zapisany w formacie XML.
- **korpus/ann_morphosyntax.xml**
Fragment korpusu z pliku `text.xml` po otagowaniu.

Dodatek C

Zawartość załączonej płyty CD

1. Niniejsza praca
2. Licencja GPL
3. Plik results.ods
4. Próbkę korpusu:
 - (a) korpus/text.xml
 - (b) korpus/ann_morphosyntax.xml
5. Pliki tekstowe:
 - (a) readme
 - (b) results.txt
 - (c) machines
 - (d) machinesAll
6. Programy napisane w języku Python:
 - (a) corpusInfo.py
 - (b) createCorpusSets.py
 - (c) example.py
 - (d) learnSelectedModels.py
 - (e) maxEnt.py
 - (f) postDistributed.py
 - (g) preDistributed.py
 - (h) showErrors.py
7. Programy napisane w języku Bash:
 - (a) chooseOkMachines.sh
 - (b) kill.sh
 - (c) ps.sh
 - (d) runDistributed.sh
 - (e) testMachines.sh

Bibliografia

- [Acedański 2010] Szymon Acedański, *A Morphosyntactic Brill Tagger for Inflectional Languages*, Advances in Natural Language Processing, strony 3–14.
- [Berger et al. 1996] Adam L. Berger, Stephen A. Della Pietra, Vincent J. Della Pietra, *A Maximum Entropy approach to Natural Language Processing*, Computational Linguistics, Volume 22, strony 39–71, 1996.
- [Curran i Clark 2003] James R. Curran, Stephen Clark, *Investigating GIS and Smoothing for Maximum Entropy Taggers*, Proceedings of the 10th Meeting of the EACL, 2003.
- [Darroch i Ratcliff 1972] John Darroch, Douglas Ratcliff, *Generalized Iterative Scaling for log-linear models*, The Annals of Mathematical Statistics, Volume 43, No. 5, strony 1470–1480.
- [Della Pietra et al. 1997] Stephen A. Della Pietra, Vincent J. Della Pietra, John Lafferty, *Inducing Features of Random Fields*, IEEE Transactions Pattern Analysis and Machine Intelligence, Volume 19, No. 4, 1997.
- [Dębowski 2001] Łukasz Dębowski, *Tagowanie i dezambiguacja morfosyntaktyczna. Przegląd metod i oprogramowania*. Nr 934, Warszawa, listopad 2001.
- [Dębowski 2004] Łukasz Dębowski, *Trigram morphosyntactic tagger for Polish*, Intelligent Information Systems, Advances in Soft Computing, Springer, strony 409–413, 2004.
- [Hajič i Hladká 1998] Jan Hajič, Barbora Hladká, *Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset*, Proceedings of COLING-ACL Conference, Montreal, Canada, strony 483–490, 1998.
- [Jurafsky i Martin 2000] Daniel Jurafsky, James H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Pearson Education, 2009.
- [Karwańska i Przepiórkowski 2009] Danuta Karwańska, Adam Przepiórkowski, *On the Evaluation of Two Polish Taggers*, The proceedings of Practical Applications in Language and Computers PALC, 2009.
- [Morfeusz] Marcin Woliński, *Analizator morfologiczny Morfeusz*, strona internetowa <http://sgjp.pl/morfeusz/>.
- [NKJP] Strona internetowa Narodowego Korpusu Języka Polskiego, <http://nkjp.pl>.
- [Piasecki i Godlewski 2006] Mariusz Piasecki, Grzegorz Godlewski, *Effective Architecture of the Polish Tagger*, TSD Volume 4188 of Lecture Notes in Computer Science, Springer, strony 213–220, 2006.

- [Przepiórkowski 2009] Adam Przepiórkowski, *A comparison of two morphosyntactic tagsets of Polish*, Representing Semantics in Digital Lexicography: Proceedings of MONDILEX Fourth Open Workshop, strony 138–144, 2009.
- [Ratnaparkhi 1996] Adwait Ratnaparkhi, *A Maximum Entropy Model for Part-Of-Speech Tagging*, Conference on Empirical Methods in Natural Language Processing (EMNLP), strony 133–142, 1996.
- [Rosenfeld 1994] Ronald Rosenfeld, *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*, PhD thesis at Carnegie Mellon University, 1994.
- [Woliński 2006] Marcin Woliński, *Morfeusz — a Practical Tool for the Morphological Analysis of Polish*, Intelligent Information Processing and Web Mining, IIS:IIPWM'06 Proceedings, strony 503–512, Springer, 2006.