Opis narzędzia do ujednoznaczniania morfoskładniowego opartego na formaliźmie *Conditional Random Fields*

Jakub Waszczuk

21 stycznia 2012

Spis treści

1 Wprowadzenie

Dokument opisuje narzędzie służące do ujednoznaczniania morfoskładniowego (tager) bazujące na probabilistycznym formaliźmie *Conditional Random Fields*. W pierwszym rozdziale opiszemy ogólny zarys działania narzędzia oraz przepływ danych pomiędzy jego komponentami. Szczegóły na temat działania poszczególnych modułów zostaną podane w kolejnych rozdziałach.

Na początek wprowadzimy podstawowe pojęcia niezbędne do zrozumienia działania tagera, którego na tym etapie będziemy traktować jak czarną skrzynkę. Przez *zdanie* będziemy rozumieli, zgodnie z potocznym rozumieniem tego słowa, sekwencję *słów*. Ponieważ poruszamy się w temacie oznaczania morfoskładniowego, zakładamy że z każdym słowem powiązany jest zestaw *interpretacji morfoskładniowych* ustalony na etapie analizy morfoskładniowej. Każda interpretacja morfoskładniowa reprezentowana jest przez dwa komponenty:

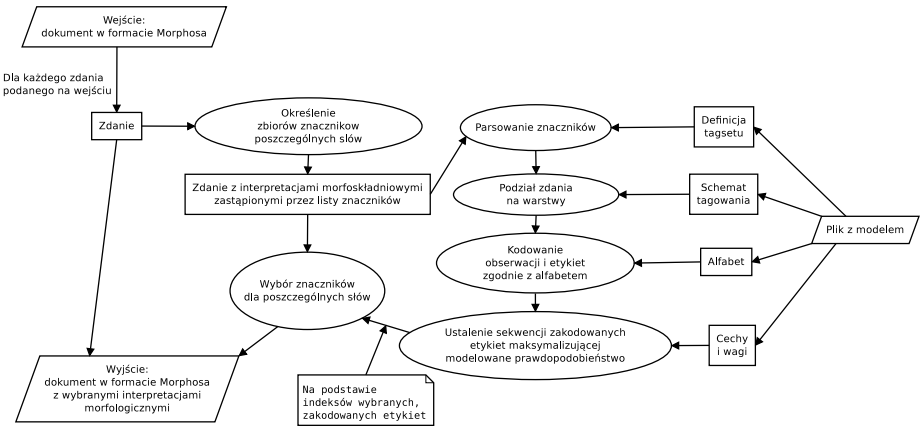
- Leksem (reprezentowany przez swoją formę podstawową), którego słowo w tekście jest wykładnikiem,
- Napis powstały przez sklejenie informacji o części mowy oraz wartościach kategorii gramatycznych. Napis ten będziemy nazywać *znacznikiem* lub *etykietą*.

W ogólnym przypadku, celem ujednoznaczniania jest określenie, m. in. na podstawie kontekstów, poprawnych interpretacji morfoskładniowych poszczególnych słów (w wyniku czego każdemu słowu przypisany zostaje dokładnie jeden leksem i znacznik). My będziemy ograniczać się jedynie do ujednoznaczniania na poziomie znaczników, czyli wyboru poprawnych interpretacji gramatycznych (wartości częsci mowy i kategorii gramatycznych) poszczególnych słów. W konsekwencji, proces ujednoznaczniania który będziemy opisywać w dalszej części tego dokumentu może prowadzić do wyboru kilku interpretacji morfoskładniowych dla niektórych słów w zdaniu (ale każda z tych interpretacji będzie identyczna na poziomie gramatycznym, czyli będzie powiązana z takim samym znacznikiem).

Ważnym etapem ujednoznaczniania jest określenie *obserwacji* poszczególnych słów. To obserwacje służą do reprezentowania słowa na poziomie modelu *Conditional Random Field* (CRF) – w trakcie trenowania tagera określane są statystyczne relacje pomiędzy etykietami określonymi dla zadanego słowa a jego obserwacjami (określane są również relacje pomiędzy sąsiadującymi etykietami, ale o tym później). Przykładowe obserwacje to: forma ortograficzna słowa (czyli słowo w swojej oryginalnej postaci), słowo poprzednie, słowo następne (lub słowo z dalszego kontekstu), a także prefiksy (pierwsze k znaków) lub sufiksy (ostatnie k znaków) formy ortograficznej. Obecna implementacja nie uwzględnia obserwacji innego typu, ale ich dodanie – jeśli okaże się potrzebne – nie powinno być trudne. Testy różnych konfiguracji tagera wykazały, że uwzględnianie słów z kontekstu ma istotny wpływ na jakość wyników ujednoznaczniania. Przykładowo, w modelu wytrenowanym w fazie wstępnych testów tagera znaleziona została następująca zależność: jeśli forma ortograficzna poprzedniego słowa jest równa „jest", to jest bardzo mało prawdopodobne, że bierzące słowo jest przymiotnikiem w bierniku. Więcej informacji na temat określania obserwacji poszczególnych słów można znaleźć w rozdziale 8.2.

Poza relacjami obserwacja-etykieta tager modeluje również relacje występujące na poziomie etykiet. Etykiety są jednak najpierw poddawane procesowi

dzielenia na warstwy, dokładniej opisanemu w rozdziale 2.4. Znaczniki morfoskładniowe składają się z wielu części reprezentujących wartości części mowy oraz różnych kategorii gramatycznych. Znaczniki są dzielone ze względu na typy kategorii i przenoszone do osobnych warstw znakowania. Przykładowy podział może wyglądać następująco: części mowy, przypadki oraz osoby trafiają do pierwszej warstwy, natomiast wartości pozostałych kategorii gramatycznych do drugiej. Weźmy przykładowe słowo „bez", dla którego analiza morfoskłaniowa daje [prep:gen:nwok, subst:pl:gen:f, subst:sg:acc:m3, subst:sg:nom:m3] (znaczniki przedstawione sa w formacie NKJP). Po przeprowadzeniu opisanego powyżej podziału każdy znacznik będzie miał postać pary, z pierwszym elementem reprezentującym wartości cześci mowy, przypadka i osoby, a drugim – wartości pozostałych kategorii. Tak więc wynik podziału będzie następujący: [(prep:gen, nwok), (subst:gen, pl:f), (subst:acc, sg:m3), (subst:nom, sg:m3)]. Tager modeluje relacje między trzema kolejnymi pod-etykietami osobno dla każdej warstwy znakowania. Tak więc, jeśli w pierwszej warstwie będą wyłącznie części mowy, a w drugiej wyłącznie przypadki, to tager osobno będzie modelował relacje między sąsiadującymi ze sobą częściamy mowy, a osobno między sąsiadującymi wartościami przypadku. Ponadto, wartości obserwacji określane są osobno nie tylko dla każdego słowa, ale także dla każdej warstwy niezależnie. Tager nie zabrania jednak zdefiniowania takich samych typów obserwacji dla różnych warstw oznaczania. Nie zabrania także przepisywania wartości jednej kategorii (podczas podziału znaczników na warstwy) do kilku różnych warstw. Sposób podziału znaczników na warstwy definiowany jest w pliku konfiguracyjnym tagera. Więcej informacji na ten temat można znaleźć w rozdziale 8.2.



Rysunek 1: Diagram przepływu danych w procesie ujednoznaczniania

Przepływ danych w trakcie ujednoznaczniania zbioru zdań wzbogaconych dodatkowymi informacjami o możliwych interpretacjach morfoskładniowych poszczególnych słów przedstawiony jest na rysunku 1. Na samym początku wczytywany jest w całości do pamięci, zapisany na dysku w postaci binarnej, model składający się z czterech komponentów. Na pierwszy z nich składają się cechy modelu wraz z odpowiadającymi im wagami, czyli właściwy model CRF. Drugim elementem jest *alfabet*, który stanowi zbiór map służących do tłumaczenia obserwacji i etykiet (które na wejściu mają postać napisów) na liczby całko-

wite, które będą reprezentować je w dalszej części procesu ujednoznaczniania. Zabieg ten ma zmiejszyć wymagania pamięciowe narzędzia i przyszpieszyć cały proces. Więcej informacji na ten temat można znaleźć w rozdziale 4. Kolejnym komponentem modelu jest *definicja tagsetu*, która pozwala parsować i poprawnie interpretować znaczniki gramatyczne. Dzięki temu możliwe jest sprawdzanie poprawności znaczników (czy są zgodne z regułami podanymi w definicji tagsetu) oraz ich „rozbijanie" na poszczególne warstwy ujednoznaczniania. Ta druga czynność jest możliwa, ponieważ w definicji tagsetu podane są również informacje o zbiorach wartości przyjmowanych przez poszczególne kategorie gramatyczne. Więcej na temat definiowania tagsetu można znaleźć w rozdziale 8.1. Ostatnim elementem jest *schemat tagowania*, główny plik konfiguracyjny tagera. W pliku tym zdefiniowane są reguły podziału na poszczególne warstwy oznaczania oraz grupy obserwacji. Więcej na temat definiowania schematu można znaleźć w rozdziale 8.2.

Po wczytaniu całego modelu rozpoczyna się proces ujednoznaczniania. Wczytywany jest – ze standardowego wejścia lub z podanego jako argument pliku – dokument w formacie Morphosa (możliwe jest również ujednoznacznianie pliku wejściowego w wewnętrznym formacie narzędzia o nazwie LINC – opis tego formatu można znaleźć w rozdziale 10). Przez format Morphosa rozumiemy tutaj format wyjściowy analizatora Morphos przeznaczony do dalszego ujednoznaczniania. Dokument wejściowy jest dzielony na zdania, zgodnie z identyfikatorami zdań, a następnie każde ze zdań poddawane jest niezależnie procesowi ujednoznaczniania. Każde ze zdań należących do zbioru wejściowego przetwarzane jest w następujący sposób:

- Dla każdego słowa określany jest zbiór potencjalnych znaczników. W wejściowym pliku w formacie Morphosa dla każdego słowa podane są wszystkie jego interpretacje morfoskładniowe. Jak już wcześniej wspomnieliśmy, wśród tych interpretacji niektóre znaczniki mogą się powtarzać, dlatego otrzymany zbiór znaczników może być mniejszy niż zbiór interpretacji morfoskładniowych.

- Znaczniki występujące w zdaniu są parsowane zgodnie z definicją tagsetu.

- Zdanie zostaje podzielone na warstwy, zgodnie ze schematem ujednoznaczniania. Dla każdego słowa określane są obserwacje należące do poszczególnych warstw oznaczania. Poszczególne znaczniki, sparsowane w poprzednim kroku, dzielone są pomiędzy poszczególne warstwy oznaczania.

- Obserwacje tłumaczone są, przy pomocy alfabetu obserwacji, na liczby całkowite. Wartości nie znalezione w alfabecie są ignorowane. Pod-etykiety (przez pod-etykietę rozumiemy tutaj połączone części pełnego znacznika, które trafiły do tej samej warstwy) tłumaczone są na liczby całkowite przy pomocy alfabetów etykiet (dla każdej warstwy zdefiniowany jest osobny alfabet). Etykieta, która nie wystąpiła w odpowiednim alfabecie, otrzymuje numer -1 (wartość nie występująca w żadnym alfabecie) – taka sytuacja jest skrajna i nie powinna wystepować.

- Algorytm opisany w rozdziale 5 zastosowany jest do określenia najbardziej prawdopodobnej sekwencji znaczników. Dokładniej, dla każdego słowa określony zostaje indeks wybranego znacznika i na tej podstawie określane są wybrane interpretacje morfoskładniowe w zbiorze wejściowym.

Wybrany znacznik może pojawiać się przy kilku interpretacjach słowa w zbiorze tekstowym – wtedy każda z tych interpretacji zostanie oznaczona jako wybrana.

Opisany powyżej proces dotyczy sytuacji ujednoznaczniania, czyli procesu w którym dla każdego słowa wybrany zostaje dokładnie jeden znacznik. Narzędzie można uruchomić również w trybie, w którym do każdego z potencjalnych znaczników przypisane jest jego prawdopodobieństwo, zgodnie z wytrenowanym modelem. Zachowana jest własność, że znacznik, który zostałby wybrany w procesie ujednoznaczniania, otrzymuje najwyższe prawdopodobieństwo. Algorytm przypisywania prawdopodobieństw również opisany jest w rozdziale 7. W sytuacji, gdy identyczny znacznik występuje w dwóch lub więcej interpretacjach morfoskładniowych formy ortograficznej występującej w tekście, prawdopodobieństwa poszczególnych interpretacji są ponownie normalizowane.

2 Conditional Random Fields

2.1 Oznaczenia

- O – Zbiór wszystkich *obserwacji*.
- Y – Zbiór wszystkich *etykiet* (lub *znaczników*), rozszerzony o specjalną etykietę γ przypisywaną niedodatnim pozycjom w zdaniu.
- $x = \{x_i\}_{i=1}^{n}$ – *Zdanie*, czyli ciąg *słów* długości n. Każde słowo x_i reprezentowane jest przez zbiór obserwacji $\{o_1, o_2, \ldots\}$, $o_i \in O$, określonych dla i-tej pozycji w zdaniu. Typowe obserwacje to: forma ortograficzna i-tego słowa, formy ortograficzne sąsiednich słów, prefiksy i sufiksy form ortograficznych itd. Obserwacje brane pod uwagę podczas trenowania i ujednoznaczniania określane są na podstawie pliku konfiguracyjnego tagera.
- $y = \{y_i\}_{i=-1}^{n}$ – Ciąg etykiet przypisanych poszczególnym słowom zdania. Dla $i \geq 1$ mamy $y_i \in Y \setminus \{\gamma\}$ oraz $y_i = \gamma$ dla $i < 1$.
- $r = \{r_i\}_{i=-1}^{n}$ – Ciąg *ograniczeń* na etykiety na poszczególnych pozycjach w zdaniu. Symbol r_i reprezentuje zbiór etykiet, które może przyjąć słowo na i-tej pozycji w zdaniu. W kontekście ujednoznaczniania, r_i reprezentuje wynik analizy morfoskładniowej i-tego słowa. Dla $i \geq 1$ mamy $r_i \subseteq Y \setminus \{\gamma\}$ oraz $r_i = \{\gamma\}$ dla $i < 1$.
- Zdanie x będziemy nazywać również *nieoznakowanym zdaniem*, natomiast parę (x, y) będziemy nazywać *oznakowanym zdaniem*, jeśli każdej pozycji i zdania x przypisana została etykieta y_i. Alternatywnie, gdy będziemy chcieli podkreślić ograniczenia na poszczególne pozycje w zdaniu, *nieoznakowanym zdaniem* będziemy nazywali parę (x, r), a trójkę (x, y, r) – *oznakowanym zdaniem*, o ile na każdej pozycji i zdania x przypisana została etykieta y_i oraz $y_i \in r_i$.
- F – Zbiór *cech* modelu CRF. Na początek założymy, że cechy modelu przyjmują jedną z dwóch postaci: (u, v, w) lub (o, u), przy czym $o \in O$

stanowi obserwację, natomiast $u, v, w \in Y$ są etykietami odpowiadającymi odpowiednio: aktualnej, poprzedniej i przed-poprzedniej pozycji w zdaniu[1]. Cechy (u, v, w) odpowiadają za modelowanie relacji między sąsiednimi etykietami, natomiast (o, u) – za modelowanie relacji między obserwacjami a etykietami określonymi względem tych samych pozycji w zdaniu. Postać cech modelu zostanie niecho zmieniona w rozdziale 2.4.

- $f_k(y, x, i)$ – Binarna funkcja f przyjmująca wartość 1 wtedy i tylko wtedy, gdy k-ta cecha modelu spełniona jest na i-tej pozycji zdania (x, y). Powiemy, że k-ta cecha modelu postaci (u, v, w) jest spełniona na i-tej pozycji oznakowanego zdania (x, y), jeśli etykiety u, v, w są równe etykietom na odpowiednich pozycjach w zdaniu, czyli $u = y_i$, $v = y_{i-1}$ oraz $w = y_{i-2}$. Analogicznie, cecha modelu postaci (o, u) jest spełniona na i-tej pozycji oznakowanego zdania (x, y), jeśli obserwacja o należy do zbioru obserwacji określonej dla i-tej pozycji zdania oraz $u = y_i$.

- $\Lambda = \{\lambda_k\}_{k=1}^{|F|}$ – *Parametry* modelu CRF. Parametr k-ty określna wagę k-tej cechy modelu (zob. wzór 1).

- $M = (F, f, \Lambda)$ – Model CRF określony przez zbiór cech F, binarną funckja f oraz zbiór parametrów Λ.

- p – Rozkład prawdopodobieństwa określony przez model CRF.

2.2 Definicja

Niech x będzie zdaniem długości n oraz y oznacza ciąg etykiet długości n. Tradycyjny model CRF (F, f, Λ) definiuje warunkowe prawdopodobieństwo $p(y|x, \Lambda)$ zgodnie ze wzorem:

$$p(y|x, \Lambda) = \frac{1}{Z(x, \Lambda)} \exp\left(\sum_{i,k} f_k(y, x, i)\lambda_k\right) \tag{1}$$

gdzie $f_k(y, x, i)$ to binarna funkcja przyjmująca wartość 1 wtedy i tylko wtedy, gdy k-ta cecha modelu spełniona jest na i-tej pozycji oznakowanego zdania (x, y), λ_k jest k-tym parametrem modelu, a funkcja $Z(x, \Lambda)$ (odpowiadająca za normalizację prawdopodobieństwa) zdefiniowana jest wzorem:

$$Z(x, \Lambda) = \sum_{y' \in (Y \setminus \{\gamma\})^n} \exp\left(\sum_{i,k} f_k(y', x, i)\lambda_k\right) \tag{2}$$

2.3 Wykorzystanie wyników analizy morfoskładniowej

Tradycyjny model CRF nie wykorzystuje dodatkowych informacji uzyskanych na etapie analizy morfoskładniowej. Zgodnie z takim modelem, na każdej pozycji zdania x może zostać przypisana dowolna (poza γ) etykieta ze zbioru Y.

[1] Tak skonstruowany model można nazwać modelem drugiego rzędu, ponieważ cechy modelu postaci (u, v, w) uwzględniają trzy następujące po sobie etykiety.

Analiza pozwala określić zbiory dopuszczalnych znaczników na poszczególnych pozycjach zdania. Podamy teraz zmodyfikowaną wersję CRFs, która uwzględnia te dodatkowe informacje. Niech x będzie zdaniem długości n, y oznacza ciąg etykiet długości n oraz r oznacza ograniczenia na wartości etykiet na poszczególnych pozycjach zdania x. Prawdopodobieństwo warunkowe $p(y|x,r,\Lambda)$ zdefiniujemy następująco:

$$p(y|x,r,\Lambda) = \begin{cases} 0 & \text{jeśli } y \notin \prod_i r_i \\ \frac{1}{Z(x,r,\Lambda)} \exp\left(\sum_{i,k} f_k(y,x,i)\lambda_k\right) & \text{wpp.} \end{cases} \tag{3}$$

Jeśli sekwencja etykiet y nie jest zgodna z zestawem ograniczeń (dla pewnego i, etykieta przypisana do i-tego słowa w zdaniu nie należy do zbioru interpretacji tego słowa określonego na etapie analizy), to modelowane prawdopodobieństwo jest równe 0. W przeciwnym razie prawdopodobieństwo obliczane jest tak jak wcześniej, z tą różnicą, że wyliczenie składnika normalizującego $Z(x,r,\Lambda)$ sprowadza się do sumowania wyłącznie po poprawnych (zgodnych z ograniczeniami) sekwencjach etykiet:

$$Z(x,r,\Lambda) = \sum_{y' \in \prod_i r_i} \exp\left(\sum_{i,k} f_k(y',x,i)\lambda_k\right) \tag{4}$$

2.4 Podział na warstwy

W zastosowaniu opisywanej metody do ujednoznaczniania morfoskładniowego liczba różnych znaczników (rozmiar zbioru Y) może być większa niż 10^3. W związku z tym liczba cech modelu, mimo że ograniczona do cech występujących w zbiorze treningowym, może być bardzo duża (jeśli przyjmiemy, że liczba różnych znaczników jest równa 10^3, to liczba wszystkich możliwych cech modelu postaci (u,v,w) wynosi 10^9).

W celu ograniczenia liczby parametrów oraz poprawienia jakości modelu wprowadziliśmy kolejną modyfikacje formalizmu CRFs. Etykiety, a jednocześnie znaczniki morfoskładniowe, składają się z wielu części reprezentujących wartości części mowy oraz różnych kategorii gramatycznych. Znaczniki są dzielone ze względu na typy kategorii i przenoszone do osobnych warstw znakowania. Fragmenty, które trafiły do tej samej warstwy, sklejane są w jedną *pod-etykietę* (lub *pod-znacznik*). Przykładowy podział może być następujący: części mowy, przypadki oraz osoby trafiają do pierwszej warstwy, natomiast wartości pozostałych kategorii gramatycznych do drugiej[2]. Sposób podziału specyfikuje się w pliku konfiguracyjnym tagera[3].

W efekcie podziału otrzymujemy L warstw o numerach $1,\ldots,L$. Ciąg etykiet y obcięty do l-tej warstwy (czyli ciąg pod-etykiet otrzymanych poprzez sklejenie przepisanych do l-tej warstwy informacji na każdej pozycji zdania) będziemy oznaczać przez $y^{(j)}$, a zbiory wszystkich pod-etykiet występujących w kolejnych warstwach – przez $Y_1, Y_2, \ldots, Y_L$. Podział znaczników będziemy opisywać przy

[2] Takie rozwiązanie zostało zainspirowane narzędziem Pantera, w którym ujednoznacznianie odbywa się wieloprzebiegowo.

[3] Konfiguracja nie wyklucza przepisania wartości wybranego atrybutu – np. części mowy – do kilku warstw.

pomocy rodziny funkcji $\delta_l : Y \to Y_l$, gdzie δ_l reprezentuje funckje wycinającą z zadanego znacznika y część należącą do l-tej warstwy. Analogicznie, ciąg słów obcięty do l-tej warstwy[4] będziemy oznaczać przez $x^{(j)}$, a ciąg ograniczeń obcięty do l-tej warstwy – i-tym elementem ciągu będzie zbiór $\{\delta_j(y) : y \in r_i\}$ – przez $r^{(j)}$. Definiujemy charakterystyczne dla poszczególnych warstw cechy modelu CRF, które przyjmują jedną z dwóch postaci: (l, u, v, w) lub (l, o, u), gdzie l reprezentuje numer warstwy, o – określoną dla l-tej warstwy obserwację, natomiast $u, v, w \in Y_l$ są należącymi do l-tej warstwy etykietami przypisanymi odpowiednio do: aktualnej, poprzedniej i przed-poprzedniej pozycji w zdaniu.

Binarną funkcję f definiujemy tym razem względem zadanej warstwy o numerze j. Niech k-ta cecha modelu ma postać (l, u, v, w). Funkcja $f_k(y, x, i, j)$ przyjmuje wartość 1 (co oznacza, że k-ta cecha modelu jest spełniona w j-tej warstwie na i-tej pozycji oznakowanego zdania (x, y)) wtedy i tylko wtedy, gdy $u = y_i^{(j)}$, $v = y_{i-1}^{(j)}$, $w = y_{i-2}^{(j)}$ oraz $j = l$. Analogicznie, cecha modelu postaci (l, o, u) jest spełniona w j-tej warstwie na i-tej pozycji oznakowanego zdania (x, y), jeśli $o \in x_i^{(j)}$, $u = y_i^{(j)}$ oraz $j = l$. Wzór na prawdopodobieństwo, przy założeniu $y \in \prod_i r_i$, przyjmuje tym razem postać:

$$p(y|x, r, \Lambda) = \frac{1}{Z(x, r, \Lambda)} \exp\left(\sum_{i,j,k} f_k(y, x, i, j)\lambda_k\right) \tag{5}$$

$$Z(x, r, \Lambda) = \sum_{y' \in \prod_i r_i} \exp\left(\sum_{i,j,k} f_k(y', x, i, j)\lambda_k\right) \tag{6}$$

3 Wydobywanie cech ze zbioru danych

W ramach trenowania modelu otrzymujemy zbiór oznakowanych zdań D i na podstawie tego zbioru określamy zbiór cech modelu CRF. Zbiór cech modelu nie zmienia się w trakcie procesu trenowania, zmieniają się jedynie wagi przypisane do poszczególnych cech. Znalezienie zbioru cech na podstawie zbioru danych sprowadza się do zsumowania zbiorów cech określonych dla poszczególnych zdań występujących w zbiorze danych. Natomiast znalezienie zbioru wszystkich cech wystepujących w zdaniu sprowadza się do zsumowania zbiorów cech określonych względem poszczególnych pozycji w zdaniu. Tak więc pozostaje opracowanie metody wydobywania cech względem i-tej pozycji oznakowanego zdania (x, y, r). W dalszej części tego rozdziału zdefiniujemy dwie metody służące do tego celu. Pierwsza z nich będzie służyła do określania cech, które – zgodnie z definicją funkcji f modelu – są spełnione na zadanej pozycji. Druga będzie służyła do określenia wszystkich cech, które mogłyby by wystąpić na zadanej pozycji, pod warunkiem że oznakowanie zdania y przyjmuje dowolną, zgodną z ograniczeniami r formę.

Dla przypomnienia, cechy modelu CRF mają jedną z dwóch postaci: (l, o, u) lub (l, u, v, w), gdzie l reprezentuje numer warstwy, $o \in O$ – obserwację, natomiast $u, v, w \in Y_l$ są należącymi do l-tej warstwy etykietami przypisanymi odpowiednio do: aktualnej, poprzedniej i przed-poprzedniej pozycji w zdaniu. Zbiór cech postaci (l, o, u) występujących na i-tej pozycji j-tej warstwy zdania

[4] Konfiguracja pozwala na definiowanie różnych typów obserwacji dla różnych warstw.

(x, y, r) można opisać wzorem $\{(j, o, y_i^{(j)}) : o \in x_i^{(j)}\}$. Pod-etykieta $y_i^{(j)}$ ustalona jest zgodnie z oznakowaniem zdania, natomiast obserwacja o może przyjąć dowolną wartość ze zbioru $x_i^{(j)}$. Jeśli chodzi o cechy postaci (l, u, v, w), to istnieje dokładnie jedna cecha tej postaci która jest – zgodnie z funckją f – spełniona na i tej pozycji j-tej warstwy zdania (x, y, r): $(j, y_i^{(j)}, y_{i-1}^{(j)}, y_{i-2}^{(j)})$. Zbiór wszystkich cech (obu postaci) występujących na i-tej pozycji oznakowanego zdania (x, y) bedziemy oznaczać przez $\mathcal{F}_i(x, y)$:

$$\mathcal{F}_i(x, y) = \bigcup_{l=1}^{L} \{(l, o, y_i^{(l)}) : o \in x_i^{(l)}\} \cup \{(l, y_i^{(l)}, y_{i-1}^{(l)}, y_{i-2}^{(l)})\} \tag{7}$$

Powyższy wzór, chociaż przydatny w innych sytuacjach, nie jest odpowiedni do konstruowania zbioru cech modelu na podstawie zbioru treningowego. W zbiorze cech modelu powinny znaleźć się wszystkie cechy, które mogłyby wystąpić w zbiorze danych, niezależnie od etykietowania poszczególnych zdań. Za tą decyzją stoi silna motywacja – w trakcie trenowania cechy, które zgodnie ze wzorem 7 nie występują w zbiorze treningowym, mogą otrzymać niezerową wagę i mieć istotny wpływ na modelowanie prawdopodobieństwa. Przyczyna jest taka, że w trakcie trenowania obliczana jest oczekiwana liczba wystąpień poszczególnych cech modelu w zbiorze danych, a ta jest niezerowa dla wszystkich cech które mogłyby w tym zbiorze wystąpić. Zbiór wszystkich cech, które mogłyby – zgodnie ze zbiorem ograniczeń r – wystąpić na i-tej pozycji zdania (x, r) bedziemy oznaczać przez $\mathcal{F}'_i(x, r)$.

$$\mathcal{F}'_i(x, r) = \bigcup_{l=1}^{L} \{(l, o, u) : o \in x_i^{(l)}, u \in r_i^{(l)}\} \cup \{(l, u, v, w) : o \in x_i^{(l)}, u \in r_i^{(l)}, v \in r_{i-1}^{(l)}, w \in r_{i-2}^{(l)}\} \tag{8}$$

Podsumowując, wzór służący do skonstruowania zbioru F wszystkich cech modelu na podstawie zbioru danych D można zapisać następująco:

$$F = \bigcup_{(x,y,r) \in D} \bigcup_{i=1}^{|x|} \mathcal{F}'_i(x, r) \tag{9}$$

4 Reprezentacja modelu

Zgodnie z wcześniej przedstawioną matematyczną definicją, na model CRF składają się następujące elementy: zbiór cech modelu (postaci (l, u, v, w) oraz (l, o, u)), funckja binarna f określająca, czy w danej wartswie na danej pozycji w zdaniu występuje podana cecha, oraz zbiór parametrów Λ reprezentujących wpływ poszczególnych cech na modelowane prawdopodobieństwo p.

Zgodnie ze wzorem 5, aby obliczyć warunkowe prawdopodobieństwo $p(y|x, r, \Lambda)$ (pomijając wyliczanie składnika normalizującego $Z(x, r, \Lambda)$) należy przeprowadzić sumowanie po wszystkich cechach modelu CRF, których może być nawet kilka milionów. Co więcej, sumowanie to odbywa się dla każdej warstwy i dla każdej pozycji w zdaniu.

Aby uniknąć tego problemu, model CRF można reprezentować jako mapę (tablicę asocjacyjną), w której cechy modelu stanowią klucze, a wagi odpowiadające poszczególnym cechom – wartości mapy. Taka reprezentacja pozwala na

szybkie wyszukiwanie wagi zadanej cechy modelu (dokładna złożoność zależy od wykorzystanej implementacji mapy). W celu obliczenia prawdopodobieństwa warunkowego p zmieniamy kolejność sumowania – najpierw znajdujemy wszystkie cechy modelu spełnione względem pozycji i (czyli, zgodnie z poprzednim podrozdziałem, $\mathcal{F}_i(x, y)$), a następnie sumujemy wagi znalezionych cech wykorzystując tablicę asocjacyjną.

Dla modelu $M = (F, f, \Lambda)$, funkcję przypisującą poszczególnym cechom modelu odpowiadajace im wartości również oznaczymy symbolem M. Funkcja ta przyjmuje wartość 0 dla cech nie należących do zbioru F, a dla należących – wartość określoną przy pomocy tablicy asocjacyjnej. Korzystając z tej notacji, wzór na prawdopodobieństwo (przy założeniu $y \in \prod_i r_i$) przyjmuje postać:

$$p(y|x, r, \Lambda) = \frac{1}{Z(x, r, \Lambda)} \exp\left(\sum_i \sum_{c \in \mathcal{F}_i(x,y)} M(c)\right) \tag{10}$$

Jest jeszcze jeden techniczny aspekt reprezentacji modelu CRF w pamięci komputera. Przyjmujemy, że etykiety oraz obserwacje należące do poszczególnych warstw reprezentowane są przez liczby całkowite $\{0, 1, 2, \ldots\}$, natomiast do przekształcania etykiet oraz obserwacji na liczby wykorzystujemy osobne mapy, konstruowane podczas wydobywania cech ze zbioru treningowego we wstępnej fazie trenowania modelu. Dokładniej, konstruujemy jedną mapę dla wszystkich obserwacji (niezależnie od tego, do której warsty należą), oraz L map dla każdej z warstw etykiet. Komplikuje to nieco implentację tagera, ale za to wyraźnie zmiejsza jego wymagania pamięciowe.

5 Ujednoznacznianie morfoskładniowe

Niech (x, r) będzie nieoznakowanym zdaniem długości n o określonych ograniczeniach r. Ujednoznacznienie polega na wybraniu najlepszych interpretacji $\hat{y}$ poszczególnych słów spośród wszystkich możliwych ciągów interpretacji $\prod_i r_i$ określonych na etapie analizy morfoskładniowej. W celu przeprowadzenia ujednoznacznienia zdania (x, r) przy wykorzystaniu modelu CRF o parametrach Λ możemy znaleźć najbardziej prawdopodobny ciąg znaczników spełniających zadane ograniczenia:

$$\hat{y} = \underset{y \in \prod_i r_i}{\arg\max}\ p(y|x, r, \Lambda) \tag{11}$$

Obliczenie wyniku powyższego wzoru wymaga wyliczenia prawdopodobieństwa względem wszystkich poprawnych (spełniających ograniczenia r) sekwencji etykiet, co w ogólnym przypadku jest bardzo czasochłonnym zadaniem. Podamy teraz szybszy algorytm, który realizuje to zadanie.

Niech x będzie ustalonym zdaniem oraz $M = (F, f, \Lambda)$ ustalonym modelem. Wprowadzimy teraz dodatkowe oznaczenie, $\phi_i(u, v, w)$, które reprezentuje sumę wag modelu M na i-tej pozycji zdania x przy założeniu, że trzy ostatnie etykiety y_i, y_{i-1} oraz y_{i-2} mają wartości, odpowiednio, u, v i w.

$$\phi_i(u, v, w) = \sum_j \left(M((j, \delta_j(u), \delta_j(v), \delta_j(w))) + \sum_{o \in x_i^{(j)}} M((j, o, \delta_j(u))) \right) \tag{12}$$

Warto zauważyć, że korzystając z nowego oznaczenia oraz z faktu, że cechy modelu mają lokalny zakres (biorą pod uwagę co najwyżej trzy sąsiadujące etykiety), wzór 11 można (dla ustalonego x) przekształcić do następującej postaci:

$$\begin{aligned}\hat{y} &= \underset{y \in \prod_i r_i}{\arg\max} \frac{1}{Z(x,r,\Lambda)} \exp\left(\sum_i \sum_{c \in \mathcal{F}_i(x,y)} M(c)\right) \\ &= \underset{y \in \prod_i r_i}{\arg\max} \sum_i \sum_{c \in \mathcal{F}_i(x,y)} M(c) \\ &= \underset{y \in \prod_i r_i}{\arg\max} \sum_i \phi_i(y_i, y_{i-1}, y_{i-2})\end{aligned} \tag{13}$$

Przez rozbicie funkcji $\arg\max$ względem poszczególnych pozycji zdania i przesunięcia symboli $\arg\max$ jak najbardziej na prawo otrzymujemy:

$$\begin{aligned}\hat{y} = \underset{y_1 \in r_1}{\arg\max}\, \phi_1(y_1, \gamma, \gamma) + \underset{y_2 \in r_2}{\arg\max}\, \phi_2(y_2, y_1, \gamma) \\ + \underset{y_3 \in r_3}{\arg\max} \cdots + \underset{y_n \in r_n}{\arg\max}\, \phi_n(y_n, y_{n-1}, y_{n-2})\end{aligned} \tag{14}$$

Na podstawie powyższego wzoru można skonstruować rekurencyjną definicję, która pozwoli w szybki sposób obliczyć najbardziej prawdopodobną sekwencję etykiet $\hat{y}$. Zdefiniujemy funckcję $\omega_i(v, w)$, która reprezentuje częściowy wynik funkcji opisanej wzorem 14 od i-tej do ostatniej pozycji w zdaniu pod warunkiem, że etykieta o numerze $i-1$ przyjmuje wartość v, a etykieta o numerze $i-2$ – wartość w.

$$\omega_{n+1}(v, w) = (0, []) \tag{15}$$

$$\omega_i(v, w) = \max_{u \in r_i} (\phi_i(u, v, w), u) \oplus \omega_{i+1}(u, v) \tag{16}$$

Symbol [] oznacza pustą listę, pary porównujemy zgodnie z porządkiem leksykograficznym, a operator $\oplus$ zdefiniowany jest wzorem:

$$(v, x) \oplus (v', xs) = (v + v', x : xs) \tag{17}$$

Gdzie $x : xs$ oznacza przyłączenie elementu x na początek listy xs. Do obliczenia poszcególnych wartości rekurencyjnej funkcji $\omega_i(v, w)$ można wykorzystać programowanie dynamiczne lub inną pokrewną metodę. Ostateczny wynik – najbardziej prawdopodobna sekwencja etykiet – przedstawia się wzorem:

$$\hat{y} = \omega_1(\gamma, \gamma) \tag{18}$$

6 Trenowanie

Trenowanie modelu CRF odbywa się na zbiorze danych T zwanym *zbiorem treningowym*. T jest zbiorem danych, czyli zbiorem oznakowanych zdań postaci (x, y, r). Wytrenowanie modelu $M = (F, f, \Lambda)$ składa się z dwóch faz: najpierw należy określić zbiór cech modelu F, a następnie wartości parametrów Λ odpowiadających poszczególnym cechom. Ostateczna postać funkcji f została zdefiniowana w rozdziale 2.4 i jest niezależna od zbioru danych. Zbiór cech konstruuje się na podstawie zbioru T przy użyciu metody opisanej w rozdziale 3. Pozostaje określenie wartości parametrów Λ modelu M.

6.1 Funkcja wiarygodności

Celem trenowania jest znalezienie takich wartości parametrów, które maksymalizują dopasowanie modelu do zbioru treningowego. Często stosowaną dla modeli CRF miarą dopasowania parametrów do zbioru treningowego jest funkcja wiarygodności $\prod_{(x,y,r)\in T} p(y|x,r,\Lambda)$. Aby uniknąć przetrenowania, miara ta regulowana jest dodatkowym parametrem $1/2\sigma^2$ pozwalającym na kontrolę absolutnych wartości parametrów modelu[5]. Ostateczny wzór na dopasowanie wygląda więc następująco:

$$\ell(\Lambda : T) = \sum_{(x,y,r)\in T} \log p(y|x,r,\Lambda) - \sum_{\lambda_k \in \Lambda} \frac{\lambda_k^2}{2\sigma^2} \tag{19}$$

Zadanie określenia wartości parametrów maksymalizujących (minimalizujących) zadaną funkcję nazywa się zadaniem *optymalizacji*, a funkcję, której wartość się maksymalizuje (minimalizuje) – *funkcją celu*. Do rozwiązania tego typu problemów najcześciej stosuje się metody wykorzystujące gradient funkcji celu do stopniowego, iteracyjnego przybliżania się do poszukiwanego optimum. Jedną z takich metod jest *metoda gradientu prostego*, jednak jej zbieżność jest wolna, a poza tym metoda wymaga wyliczania pełnego gradientu funkcji celu w każdej interacji algorytmu. W sytuacji, gdy funkcja celu stanowi sumę różniczkowalnych składowych (np. sumę po wszystkich elementach zbioru treningowego, tak jak ℓ), można wykorzystać metodę o nazwie *Stochastic Gradient Descent* (SGD).

6.2 Stochastic Gradient Descent

Wykorzystując fakt, że funkcja celu stanowi sumę różniczkowalnych funkcji składowych, wyliczanie gradientu można sprowadzić do wysumowania gradientów obliczonych względem poszczególnych składowych. Ponadto, zamiast wyliczać pełny gradient funkcji celu, można na bierząco aplikować gradienty wyliczone dla kolejnych części – dzięki temu metoda SGD wykazuje się szybką zbieżnością i jest dobrze skalowalna. W przypadku funkcji ℓ stosujemy drobną modyfikację tej techniki. Wyliczanie gradientu dla pojedynczych elementów zbioru treningowego wymagałoby każdorazowo dokonania regulacji parametrów modelu (proces ten ma złożoność $\mathcal{O}(|\Lambda|)$), dlatego gradient obliczany jest względem małych, b-elementowych (np. 30-elementowych) porcji zbioru danych.

Niech B stanowi podział zbioru T na n części. Wtedy funkcję celu możemy zapisać jako:

$$\ell(\Lambda : T) = \sum_{i=1}^{n} \hat{\ell}(\Lambda : B_i) \tag{20}$$

$$\hat{\ell}(\Lambda : B_i) = \sum_{(x,y,r)\in B_i} \log p(y|x,r,\Lambda) - \frac{|B_i|}{|T|} \sum_{\lambda_k \in \Lambda} \frac{\lambda_k^2}{2\sigma^2} \tag{21}$$

Poszukiwanie optimum rozpoczynamy od wybranego punktu Λ_0 (w którym np. wszystkie parametry są równe 0). W kolejnych iteracjach $i = 0, 1, \ldots$ losowana jest b-elementowa kombinacja z powtórzeniami B_i zbioru T (powyższy

[5] Modyfikacja ta jest równoważna nadaniu parametrom modelu normalnego rozkładu *a priori* $\mathcal{N}(0, \sigma)$. Wytrenowanie modelu polega wtedy na znalezieniu estymatorów maksymalizujących rozkład *a posteriori*.

opis metody SGD sugeruje inne podejście – zbiór T jest losowo dzielony na b-elementowe części, na kolejnych częściach obliczany jest gradient i tak w kółko.), a parametry modyfikowane są zgodnie ze wzorem:

$$\Lambda_{i+1} = \Lambda_i + \eta_i \nabla \hat{\ell}(\Lambda_i : B_i) \tag{22}$$

Współczynnik długości kolejnych kroków η_i przedstawia się wzorem:

$$\eta_i = \eta_0 \frac{\tau}{\tau + m} \tag{23}$$

Gdzie τ to stała równa liczbie przejść po całym zbiorze danych, po której współczynnik η dwukrotnie maleje, m to liczba przejść po całym zbiorze danych, czyli około $i \cdot b/|T|$, a η_0 to początkowa wartość współczynnika długości kroków. Podczas trenowania tagera stałe w opisanym wcześniej algorytmie zostały ustawione następująco: $b = 30$, $\sigma = 10.0$, $\eta_0 = 1.0$ oraz $\tau = 5.0$.

6.3 Gradient funkcji wiarygodności

W trakcie optymalizacji parametrów modelu CRF obliczany jest gradient funkcji celu $\hat{\ell}$, który wskazuje kierunek najszybszego wzrostu wartości funkcji $\hat{\ell}$. Gradient względem kombinacji B zbioru T obliczany jest zgodnie ze wzorem:

$$\frac{\partial \hat{\ell}(\Lambda : B)}{\partial \lambda_k} = N_k(B) - \mathbb{E}\left[N_k(B)|\Lambda\right] - \frac{|B|}{|T|} \cdot \frac{\lambda_k}{\sigma^2} \tag{24}$$

Gdzie $N_k(B)$ oznacza liczbę wystąpień k-tej cechy w zbiorze B, a $\mathbb{E}\left[N_k(B)|\Lambda\right]$ reprezentuje oczekiwaną liczbę wystąpień k-tej cechy w tym zbiorze.

Metoda wyszukiwania wszystkich cech, które występują w zadanym zbiorze, została już wspomniana w rozdziale 3. Obliczanie wartości $N_k(B)$ jest podobne – wystarczy zliczyć wystąpienia k-tej cechy w zbiorze B. Niezależnie od postaci k-tej cechy modelu, obliczanie wartości $N_k(B)$ sprowadza się do wysumowania liczb wystąpień k-tej cechy w poszczególnych zdaniach zbioru B na poszczególnych pozycjach tych zdań:

$$N_k(B) = \sum_{(x,y,r) \in B} \sum_{i=1}^{|x|} n_k(x, y, r, i) \tag{25}$$

Jeśli k-ta cecha modelu ma postać (l, o, u), gdzie l to numer warstwy, o to obserwacja a u to pod-etykieta należąca do l-tej warstwy, to wzór na $n_k(x, y, r, i)$ przedstawia się następująco:

$$n_k(x, y, r, i) = [o \in x_i^{(j)} \wedge u = y_i^{(j)}] \tag{26}$$

Natomiast dla cech modelu postaci (l, u, v, w), gdzie l to numer warstwy, a u, v, w to etykiety z l-tej warstwy:

$$n_k(x, y, r, i) = [u = y_i^{(j)} \wedge v = y_{i-1}^{(j)} \wedge w = y_{i-2}^{(j)}] \tag{27}$$

Gdzie $[\cdot]$ to nawias Iversona (wartość wyrażenia jest równa 1, jeśli warunek w nawiasie jest spełniony, wpp. 0).

W praktyce obliczenia te lepiej jest przeprowadzać w innej kolejności niż wynika to z matematycznych wzorów. Dla zadanego zbioru oznakowanych zdań B należy przeszukać wszystkie zdania tego zbioru na wszystkich pozycjach i korzystając ze wzoru 7 na bierząco aktualizować informacje o liczbie wystapień poszczególnych cech w (początkowo pustej) strukturze danych, np. mapie.

6.4 Oczekiwana liczba wystąpień cechy w zbiorze danych

Obliczanie oczekiwanej liczby wystapień $\mathbb{E}[N_k(B)|\Lambda]$ k-tej cechy w zbiorze B jest trudniejsze niż proste wyliczenie liczby wystąpień $N_k(B)$. W trakcie obliczania oczekiwanej liczby $N_k(B)$ ignorujemy etykietowania y poszczególnych zdań w zbiorze B, a oczekiwaną liczbę ustalamy na bazie parametrów Λ modelu.

Niech $\mathbb{Y}$ oznacza zmienną losową reprezentującą sekwencję etykiet (w przeciwieństwie do y, która oznacza ciąg ustalonych etykiet, czyli wartość zmiennej $\mathbb{Y}$; $p(y|x,r,\Lambda)$ można inaczej zapisać jako $p(\mathbb{Y}=y|x,r,\Lambda)$). Korzystając z tego że wartość oczekiwana sumy jest równa sumie wartości oczekiwanych, możemy (podobnie jak przy obliczaniu $N_k(B)$) napisać:

$$\mathbb{E}[N_k(B)|\Lambda] = \sum_{(x,y,r)\in B} \sum_{i=1}^{|x|} \mathbb{E}[n_k(x,y,r,i)|\Lambda] \tag{28}$$

Dla cech postaci (l,o,u), gdzie l to numer warstwy, o to obserwacja a u to podetykieta należąca do l-tej warstwy, wzór na $\mathbb{E}[n_k(x,y,r,i)|\Lambda]$ można zapisać tak:

$$\mathbb{E}[n_k(x,y,r,i)|\Lambda] = p(\mathbb{Y}_i^{(j)} = u|x,r,\Lambda)\cdot[o\in x_i^{(j)}] \tag{29}$$

Natomiast dla cech modelu postaci (l,u,v,w), gdzie l to numer warstwy, a u, v, w to etykiety z l-tej warstwy:

$$\mathbb{E}[n_k(x,y,r,i)|\Lambda] = p(\mathbb{Y}_i^{(j)} = u, \mathbb{Y}_{i-1}^{(j)} = v, \mathbb{Y}_{i-2}^{(j)} = w|x,r,\Lambda) \tag{30}$$

Do tego:

$$p(\mathbb{Y}_i^{(j)} = u, \mathbb{Y}_{i-1}^{(j)} = v, \ldots|x,r,\Lambda) = \sum_{u'\in r_i, v'\in r_{i-1},\ldots} p(\mathbb{Y}_i = u', \mathbb{Y}_{i-1} = v', \ldots|x,r,\Lambda)\cdot[\delta_l(u') = u, \delta_l(v') = v, \ldots] \tag{31}$$

Wartość $p(\mathbb{Y}_i = u', \mathbb{Y}_{i-1} = v', \ldots|x,r,\Lambda)$ można obliczyć wprost z definicji $p(y|x,r,\Lambda)$ ze wzoru na rozkład brzegowy, ale poniżej podamy dużo szybszą metodę (zob. wzory 38, 39, 40).

Podobnie jak w przypadku wyliczania rzeczywistej liczby wystąpnień zadanej cechy w zbiorze danych, tutaj również kolejność obliczeń w implementacji tagera jest odwrotna. Dla każdego zdania (x,y,r) w zbiorze danych oraz względem każdej pozycji i tego zdania należy znaleźć (korzystając ze wzoru 8 na $\mathcal{F}_i'(x,r)$) cechy mające niezerowe prawdopodobieństwo wystąpienia w danym miejscu i zaktualizować informacje w odpowiedniej strukturze danych.

6.5 Tablice „w przód" i „w tył"

Tablice „w przód" i „w tył" (od angielskiej nazwy algorytmu *forward-backward algorithm*), które w tym rozdziale będziemy oznaczać przez α i β, służą (między innymi) do reprezentowania częściowych wyników otrzymywanych w trakcie obliczania wartości $Z(x,r,\Lambda)$ dla zadanego zdania (x,r) długości n i parametrów modelu Λ. Niech $\hat{\phi}_i(u,v,w) = \exp\phi_i(u,v,w)$. Korzystając z analogicznych

przekształceń jak we wzorach 13 oraz 14 otrzymujemy następujący wzór:

$$Z(x,r,\Lambda) = \sum_{y_1 \in r_1} \hat{\phi}_1(y_1,\gamma,\gamma) \sum_{y_2 \in r_2} \hat{\phi}_2(y_2,y_1,\gamma) \cdot \sum_{y_3 \in r_3} \cdots \sum_{y_n \in r_n} \hat{\phi}_n(y_n,y_{n-1},y_{n-2}) \tag{32}$$

Powyższy wzór można zamienić na rekurencyjną definicję, która pozwoli szybko obliczać – np. przy wykorzystaniu programowania dynamicznego – wartość funkcji $Z(x,r,\Lambda)$ (por. ze wzorem 15). Wartość funckji $\alpha_i(u,v)$ reprezentuje częściowy wynik $Z(x,r,\Lambda)$ do i-tej pozycji w zdaniu (wartości funkcji $\hat{\phi}_j$ dla $j > i$ są ignorowane) przy założeniu, że etykieta o numerze $i-1$ przyjmuje wartość v, a etykieta o numerze i – wartość u.

$$\alpha_0(u,v) = 1 \tag{33}$$

$$\alpha_i(u,v) = \sum_{w \in r_{i-2}} \hat{\phi}_i(u,v,w) \cdot \alpha_{i-1}(v,w) \text{ dla } i > 0 \tag{34}$$

Warto zauważyć, że:

$$Z(x,r,\Lambda) = \sum_{(u,v) \in r_n \cdot r_{n-1}} \alpha_n(u,v) \tag{35}$$

Definicja tablicy β, która przechowuje częściowe wyniki obliczania $Z(x,r,\Lambda)$ „od tyłu" jest symetryczna. Wartość $\beta_i(v,w)$ reprezentuje częściowy wynik $Z(x,r,\Lambda)$ od n-tej do i-tej pozycji w zdaniu (wartości funkcji $\hat{\phi}_j$ dla $j < i$ nie są brane pod uwagę) przy założeniu, że etykieta o numerze $i-1$ przyjmuje wartość v, a o numerze $i-2$ – wartość w. Rekurencyjna definicja β wygląda następująco:

$$\beta_{n+1}(v,w) = 1 \tag{36}$$

$$\beta_i(v,w) = \sum_{u \in r_i} \hat{\phi}_i(u,v,w) \cdot \beta_{i+1}(u,v) \text{ dla } i < n+1 \tag{37}$$

Dzięki tablicom α i β można nie tylko szybko obliczyć wartość $Z(x,r,\Lambda)$, ale również wartości prawdopodobieństw które pojawiły się w trakcie wyliczania gradientu (zob. wzór 31). Niech α i β będą tablicami wyliczonymi względem zdania (x,r) i parametrów Λ oraz niech $u \in r_i$, $v \in r_{i-1}$, $w \in r_{i-2}$.

$$p(\mathbb{Y}_i = u, \mathbb{Y}_{i-1} = v, \mathbb{Y}_{i-2} = w | x,r,\Lambda) = \alpha_{i-1}(v,w) \cdot \hat{\phi}(u,v,w) \cdot \beta_{i+1}(u,v) \;/\; Z(x,r,\Lambda) \tag{38}$$

$$p(\mathbb{Y}_i = u, \mathbb{Y}_{i-1} = v | x,r,\Lambda) = \alpha_i(u,v) \cdot \beta_{i+1}(u,v) \;/\; Z(x,r,\Lambda) \tag{39}$$

$$p(\mathbb{Y}_i = u | x,r,\Lambda) = \sum_{v \in r_{i-1}} p(\mathbb{Y}_i = u, \mathbb{Y}_{i-1} = v | x,r,\Lambda) \tag{40}$$

6.6 Dokładność obliczeń

Wartości otrzymywane w trakcie trenowania modelu mogą być zarówno bardzo duże, jak i bardzo małe, co może skutkować ich „wypadnięciem" poza zakres

liczb zmiennoprzecinkowych. Przykładowo, wartość $Z(x, r, \Lambda)$ dla danego zdania (x, r) i parametrów Λ może być ogromna, natomiast wartość prawdopodobieństwa $p(y|x, r, \Lambda)$ – bardzo bliska 0. W celu uniknięcia błędów wynikających z niedokładnej reprezentacji liczb rzeczywistych w pamięci komputera, część obliczeń opisanych w rozdziale o trenowaniu modelu odbywa się w skali logaryczmicznej. Dokładniej, w skali logaryczmicznej obliczana jest oczekiwana wartość $\mathbb{E}\left[N_k(B)|\Lambda\right]$ liczby wystąpień k-tej cechy modelu w zbiorze danych B. Wszystkie obliczenia z tym związane – np. konstruowanie tablic α i β – również przeprowadzane są w skali logarytmicznej. Dopiero bezpośrednio przed włączeniem oczekiwanej wartości do gradientu $\nabla\hat{\ell}$ (zob. wzór 24) jest ona przekształcana (przez zaaplikowanie funkcji exp) do normanej skali.

7 Oznaczanie prawdopodobieństwami

Proces *oznaczania prawdopodobieństwami* jest bardzo podobny do ujednoznaczniania opisanego w rozdziale 5. Różnica polega na tym, że zamiast wybierać „najlepszy" znacznik dla każdego słowa w zdaniu, chcemy otrzymać prawdopodobieństwa wszystkich potencjalnych znaczników poszczególnych słów w zdaniu.

Jako pierwsze rozwiązanie do głowy przychodzi przypisywanie poszczególnym etykietom prawdopodobieństw z rozkładów brzegowych, czyli prawdopodobieństw określonych wzorem 40. Rozwiązanie to ma jedną wadę, a mianowicie nie zachowuje następującej właśności: znacznik, który zostałby wybrany w procesie ujednoznaczniania, otrzymuje w procesie oznaczania prawdopodobieństwami najwyższe prawdopodobieństwo spośród wszystkich interpretacji danego słowa[6]

Opiszemy teraz rozwiązanie, które spełnia podaną właśność. Dla zadanego zdania (x, r), słowa x_i oraz potencjalnej etykiety $r_{i,j}$ będziemy obliczać sekwencję etykiet $\hat{y}$ zgodną ze wzorem:

$$\hat{y} = \underset{y \in \prod_i r_i, \hat{y}_i = r_{i,j}}{\arg\max} p(y|x, r, \Lambda) \tag{41}$$

Czyli chcemy znaleźć sekwencję etykiet maksymalizującą modelowane prawdopodobieństwo przy założeniu, że $\hat{y}_i = r_{i,j}$. Przyjmiemy, że prawdopodobieństwo etykiety $r_{i,j}$ jest równe prawdopodobieństwu otrzymanej sekwencji etykiet, czyli $p(\hat{y}|x, r, \Lambda)$. Ponieważ taka definicja nie gwarantuje, że prawdopodobieństwa etykiet ze zbioru r_i będą się sumować do 1, należy je znormalizować dla każdego słowa w zdaniu.

Po przeczytaniu powyższej definicji może się wydawać, że oznaczenie prawdopodobieństw dla wszystkich słów w zdaniu jest bardzo czasochłonne, ponieważ wymaga obliczenia wzoru podobnego do wzoru na ujednoznacznianie tyle razy, ile jest potencjalnych etykiet w zdaniu. W praktyce jednak istnieje rozwiązujący ten problem algorytm o złożoności czasowej takiej samej jak algorytm rozwiązujący problem ujednoznaczniania.

Wykorzystamy definicje tablic α i β z poprzedniego rozdziału zmodyfikowane na potrzeby oznaczania prawdopodobieństwami. Podamy od razu rekurencyjne

[6] Rozwiązanie to zostało jednak zaimplementowane i wydaje się, że w praktyce daje równie dobre wyniki jak rozwiązanie spełniające podaną własność.

wzory:

$$\alpha'_0(u,v) = 1 \tag{42}$$

$$\alpha'_i(u,v) = \max_{w \in r_{i-2}} \hat{\phi}_i(u,v,w) \cdot \alpha_{i-1}(v,w) \text{ dla } i > 0 \tag{43}$$

$$\beta'_{n+1}(v,w) = 1 \tag{44}$$

$$\beta'_i(v,w) = \max_{u \in r_i} \hat{\phi}_i(u,v,w) \cdot \beta_{i+1}(u,v) \text{ dla } i < n+1 \tag{45}$$

Jedyną różnicą w stosunku do wcześniejszych definicji jest zamiana funkcji $\sum$ na max. Wartość $\alpha'_i(u,v)$ jest równa maksymalnemu, częściowemu (od pozycji 1 do pozycji i) prawdopodobieństwu (nieznormalizowanemu czynnikiem $Z(x,r,\Lambda)$) osiąganemu względem dowolnej sekwencji etykiet $y \in \prod_{k=1}^{i} r_k$ takiej, że $y_i = u$ oraz $y_{i-1} = v$. Analogicznie, $\beta'_i(v,w)$ reprezentuje maksymalne cześciowe (od pozycji i do pozycji n) nieznormalizowane prawdopodobieństwo osiągane względem dowolnej sekwencji etykiet $y \in \prod_{k=i-2}^{n} r_k$ takiej, że etykieta o numerze $i-1$ przyjmuje wartość v, a o numerze $i-2$ – wartość w.

Warto zwrócić uwagę na podobieństwo pomiędzy wzorem na funkcję β' a wzorem rekurencyjnym na funkcję ω zdefiniowanym w rozdziale 5 dotyczącym ujednoznaczniania. W rzeczywistości wzory te obliczają niemalże to samo, z tą różnicą że obliczenia ω przeprowadzane są *explicite* w skali logarytmicznej oraz dają w rezultacie nie tylko wartość prawdopodobieństwa, ale również sekwencję etykiet która to prawdopodobieństwo maksymalizuje.

Przy wykorzystaniu powyższych pojęć można już obliczyć szukane prawdopodobieństwo etykiety u na pozycji i zdania (x,r) zdefiniowane wcześniej wzorem 41:

$$\max_{v \in r_{i-1}} \alpha'_i(u,v) + \beta'_{i+1}(u,v) \tag{46}$$

Na koniec uwaga techiczna: tak jak w przypadku trenowania, tak i tutaj większość obliczeń wykonywana jest w skali logaryczmicznej.

8 Konfiguracja

Są dwie główne metody konfiguracji tagera. Pierwsza z nich dotyczy dostosowywania narzędzia do wykorzystywanego w danych wejściowych tagsetu. Druga cześć konfiguracji pozwala zmieniać parametry ujednoznaczniania, takie jak liczba warstw oznaczania oraz brane pod uwagę typy obserwacji. Jest jeszcze trzecia metoda konfiguracji tagera, ale dotyczy ona wyłącznie fazy trenowania i nie będziemy się nią w tym rozdziale zajmować.

8.1 Definicja tagsetu

Definicja tagsetu zapisana jest w zewnętrznym pliku, co pozwala na wykorzystywanie tagera do ujednoznaczniania różnych zbiorów danych. Różnice pomiędzy tagsetami mogą występować na dwóch poziomach:

- Formatu tagów, czyli w sposobie zapisu informacji o wartościach części mowy i kategorii gramatycznych,

- Zakresu informacji, które są reprezentowane na poziomie znaczników.

Obecnie tager wspiera dwa sposoby zapisu znaczników – format NKJP oraz format Morphosa. W formacie NKJP poszczególne pola znaczników oddzielone są separatorem ':'. W formacie Morphosa nie ma separatorów, ale za to brak wartości opcjonalnej kategorii oznaczany jest specjalnych znakiem '_'.

Oprócz formatu zapisu znaczników, w definicji tagsetu należy określić:

- Typy atrybutów oraz przyjmowane przez nie wartości. Przez atrybut rozumiemy tutaj cześć mowy (`pos`) lub dowolną kategorie gramatyczną.
- Reguły parsowania w podziale na wartości wiodących pól znacznika (w najprostszej wersji, reguły parsowania można definiować w podziale na wartości atrybutu cześci mowy).

Prześledzimy teraz przykładową definicję tagsetu przygotowaną dla formatu Morphosa. Na początek należy określić format znaczników:

```
# Format of labels
view    = Morphos
```

Obecnie jedynie dwie wartości są dopuszczalne, `Morphos` dla znaczników w formacie Morphosa oraz `NKJP` dla znaczników w formacie NKJP.

Następnie należy wypełnić dwie sekcje definicji. Pierwsza z nich to sekcja atrybutów. W ramach sekcji atrybutów należy podać definicje atrybutu `pos` (części mowy) oraz pozostałych kategorii gramatycznych i atrybutów specjalnych branych pod uwagę:

```
# Attributes and their values
[ATTR]

# Part of speech
pos = v n abr adj proa proadv adv pronum # ...

# Verb form
vfm = A B C D E F G H I K L M N

# Grammatical categories
nmb     = s p # number
cas     = N G D A I L V # case
gnd     = o z r f n p1 p2 # gender
# ...
```

W kolejnej sekcji podaje się reguły parsowania znaczników. Sekcja ta dzieli się na podsekcje. Każda z nich rozpoczyna się od linii zawierającej informacje, które atrybuty wiodące mają determinować sposób parsowania pozostałych atrybutów. Przykładowo, gdy parsowanie ma odbywać się zależnie od wartości części mowy, defnicja podsekcji może wyglądać tak:

```
{pos}
n = cas nmb gnd
proa = cas nmb gnd
num = cas nmb gnd
```

```
adj = [cas] [nmb] [gnd] [deg]
prep =
# ...
```

Po lewej stronie znaku równości podana jest wartość atrybutu `pos`, natomiast po prawej – lista atrybotów (być może pusta), których wartości powinny pojawiać się w ramach znacznika, zgodnie z podaną kolejnością. Jeśli atrybut jest opcjonalny, powinien zostać podany w nawiasach kwadratowych.

Kolejny przykład pokazuje definicję podsekcji, w której parsowanie atrybutów zależy od wartości części mowy i formy czasownika. Taki sposób definiowania reguł wymaga, żeby wartości podane po lewej stronie znaku równości pojawiały się w polach parsowanego znacznika jako pierwsze.

```
{pos vfm}
v A = per nmb tns mod asp
v G = cas nmb gnd asp
v I = asp
# ...
```

8.2 Schemat oznaczania

Każde zdanie ze zbioru wejściowego jest dzielone na warstwy zgodnie ze schematem oznaczania. Schemat definiuje, jakie typy obserwacji brane są pod uwagę, oraz do których warstw trafiają wartości poszczególnych atrybutów określone na etapie parsowania znaczników.

Schemat składa się z dwóch sekcji. Pierwszą z nich jest sekcja obserwacji, która składa się z listy reguł łączenia prostych typów obserwacji w większe, złożone typy. Na razie zdefiniowane są następujące, atomowe typy, na bazie których można budować bardziej złożone konstrukcje:

- `@orth n` – forma ortograficzna słowa na pozycji $i + n$, gdzie i to pozycja obecnie rozpatrywanego słowa.
- `@prefix k n` – prefiks długości k słowa na pozycji $i+n$, gdzie i to pozycja obecnie rozpatrywanego słowa.
- `@suffix k n` – analogicznie zdefiniowany sufiks.

Przykładowa sekcja obserwacji może wyglądać następująco:

```
# Definicje grup obserwacji
[observations]
orth = @orth 0 # forma ortograficzna bierzącej pozycji
orth_m1 = @orth -1 # forma ortograficzna poprzedniej pozycji
orth_m2 = @orth -2

# Substytut formy podstawowej -- prefiksy różnych długości.
base = @prefix 0 0 | @prefix -1 0 | @prefix -2 0 | @prefix -3 0
prefix3_pair = @prefix 3 0 + @prefix 3 -1
suff3 = @suffix 3 0
pref3 = @prefix 3 0
```

W powyższym przykładzie do budowania złożonych typów wykorzystujemy dwa specjalne operatory:

- Symbol | reprezentuje połączenie typów składowych w jedną grupę. Gdy typ złożony uzyskany przy użyciu tego symbolu zostanie wykorzystany w wybranej warstwie oznaczania, wartości wszystkich typów składowych będą brane pod uwagę jako obserwacje wszystkich słów występujących w zdaniu. Ważny jest też fakt, że wartości poszczególnych składowych konstrukcji z operatorem | nie są rozróżniane. Zgodnie z powyższym przykładem, prefiks długości 1 słowa *koty* i prefiks długości 0 słowa *kot* stanowią tą samą obserwację. W ogólnym przypadku ta reguła nie działa – nie chcemy, żeby wartości różnych *typów* obserwacji były równe nawet wtedy, gdy ich *wartości* są równe.

- Symbol + reprezentuje połączenie wartości dwóch typów obserwacji w jedną wartość. W kontekście powyższego przykładu, wartością obserwacji `prefix3_pair` dla słowa *kot*, które jest poprzedzone w zdaniu słowem *czarny*, będą połączone wartości prefiksów długości 3 obu tych słów.

W kolejnej cześci schematu definiowane są kolejne warstwy oznaczania. Definicja każdej wartstwy stanowi osobną sekcję dokumentu, w której należy zdefiniować wartości dwóch zmiennych – `tags` oraz `observations`. W ramach pierwszej z nich należy podać typy atrybutów gramatycznych, które mają być uwzględnione w danej warstwie oznaczania. Wartość drugiej określa typy obserwacji, które w danej warstwie mają być brane pod uwagę, przy czym podając wartość tej zmiennej można odnosić się wyłącznie do obserwacji złożonych zdefiniowanych w ramach sekcji obserwacji (czyli nie można odnościć się do typów atomowych).

```
# Pierwsza warstwa oznaczania
[layer]
# część mowy, przypadek i osoba
tags = pos cas per
# obserwacje w pierwszej warstwie
observations = orth orth_m1

# Druga warstwa oznaczania
[layer]
# wszystkie pozostałe atrybuty
tags = *
# obserwacje w drugiej warstwie
observations = orth orth_m1
```

9 Użycie

9.1 Wymagania

Implementacja tagera została przygotowana w języka Haskell. Polecanym kompilatorem dla tego języka jest Glasgow Haskell Compiler (GHC). Narzędzie zostało przetestowane na wersjach GHC 7.03 i 7.04. Są dwie ścieżki instalacji kolpilatora GHC. Pierwsza, trudniejsza, wiąże się z niezależną instalacją kompilatora ze strony `http://www.haskell.org/ghc/` i dodatkowych narzędzi (np. menedżera pakietów Hakslla o nazwie cabal). Wygodniej jednak jest zainstalować *platformę Haskell-a* (`http://hackage.haskell.org/platform/`), która

razem z kompilatorem dostarcza również narzędzie cabal i szereg podstawowych bibliotek Haskell-a. W dalszej cześci tego podrozdziału zakładamy, że GHC został zainstalowany w ramach platformy Haskell-a.

Tager korzysta z kilku zewnętrznych bibliotek. Aby je zainstalować, należy wpisać następujące komendy w wierszu poleceń:

```
cabal update
cabal install binary
cabal install data-memocombinators
cabal install cmdargs
```

9.2 Kompilacja

Tager znajdujący się w katalogu src można skompilować z linii poleceń komendą:

```
ghc -O2 Tager -o tager
```

Program wynikowy zostanie zapisany w pliku 'tager'. GHC przyjmuje jako argumenty szereg istotnych flag sterujących procesem kompilacji. Przykładowo, aby tager można było uruchamiać w trybie współbieżnym, należy użyć następującej komendy:

```
ghc -O2 -rtsopts -threaded Tager -o tager
```

Dodatkowa flaga `-rtsopts` pozwoli podczas uruchamiania tagera sterować liczbą wykorzystywanych rdzeni. Poza tym czasami przydatna może być się flaga `-fforce-recomp`, która wymusza rekompilację całego kodu tagera.

Poniżej opiszemy sposoby użycia skompilowanego programu do celów tagowania, trenowania i walidacji krzyżowej. Opis trybów pracy tagera można zobaczyć uruchamiając program z włączoną flagą `--help`:

```
./tager --help
```

9.3 Uruchamianie w trybie współbieżnym

Domyślnie tager uruchamiany jest w trybie jednowątkowym – wykorzystuje tylko jeden rdzeń, niezależnie od liczby dostępnych procesorów. Aby uruchomić tager w trybie współbieżnym, należy go najpierw skompilować z włączonymi flagami `-rtsopts` i `-threaded`, co było opisane w poprzednim rozdziale. Uruchamianie w trybie współbieżnym uzyskuje się przez podanie specjalnych argumentów środowiska wykonywania programu:

```
./tager OPCJE-TAGERA +RTS -Ni
```

Wszystkie argumenty następujące po `+RTS` zostaną przekierowany do środowiska wykonywania programu. Opcja `-Ni` steruje liczbą wykorzystywanych rdzeni (należy za *i* podstawić konretną wartość, np. 4).

Tager wykorzystuje możliwości pracy współbieżnej podczas trenowania modelu (przy jednoczeńśnie podanej wartości argumentu `--workersnum`) oraz podczas tagowania plików wejściowych.

9.4 Tagowanie

Opisywane poniżej komendy działają pod systemem Linuks. Pod Windows tager należy uruchamiać w ten sam sposób, ale pomocnicze narzędzia (cat, iconv) są zainstalowane domyślnie tylko pod systemem Linuks. Tagowanie plików wyjściowych analizy morfoskładniowej w formacie Morphosa można uruchomić komendą:

```
cat input.txt | ./tager tag model -m
```

Gdzie `input.txt` to plik wejściowy (wynik analizy), a `model` to plik z wytrenowanym modelem. Opcja `-m` podaje tagerowi informację, że plik wejściowy jest w formacie Morphosa. Tagowanie można również uruchomić na zadanym opcją `-d` pliku:

```
./tager tag model -m -d input.txt
```

Powyżej zakładamy, że plik wejściowy kodowany jest w UTF-8. Jeśli jest inaczej – przykładowo, plik wejściowy kodowany jest w ISO_8859-2 – można użyć programu iconv do zmiany kodowania:

```
iconv -f ISO_8859-2 -t UTF-8 input.txt | ./tager tag model -m
```

Wynik tagowania wypisywany jest na standardowe wyjście, błędy natomiast na stderr; wyjścia można przekierować aby zapisać wyniki do plików, np:

```
... | ./tager tag -m model > output.txt 2> errors.txt
```

Można również tagować pliki wejściowe w wewnętrznym formacie LINC. Format ten jest opisany w rozdziale 10. Komendy są w takim przypadku analogiczne jak powyżej, należy jedynie pominąć flagę `-m` w argumentach programu, np.:

```
./tager tag model -d input.linc
```

Dla formatu LINC tager domyślnie działa w trybie ujednoznaczniania, natomiast dla formatu Morphosa przyporządkowuje poszczególnym interpretacjom prawdopodobieństwa.

Ogólny opis wszystkich argumentów programu w trybie tagowania można uzyskać uruchamiając narzędzie następującą komendą:

```
./tager tag --help
```

9.5 Trenowanie

Trenowanie modelu odbywa się na zbiorze danych w formacie LINC kodowanym w UTF-8. Zakładamy przy tym, że każde zdanie treningowe zapisane jest w osobnym pliku o rozszerzeniu `.linc` oraz że wszystkie znajdują się w tym samym katlogu `train`. Opcjonalnie można podać katalog z plikami do ewaluacji `eval`, które będą służyć do okresowego sprawdzania jakości modelu w trakcie procesu trenowania.

Do uruchomienia procesu trenowania służy następująca komenda:

```
./Tager train -e eval tagset.cfg schema.cfg train
```

Gdzie `tagset.cfg` to definicja tagsetu (opis przygotowania definicji tagsetu znajduje się w rozdziale 8.1), a `schema.cfg` to schemat oznaczania (opisany w rozdziale 8.2).

Do sterowania procesem trenowania służy szereg dodatkowych opcji programu. Ich ogólny opis można uzyskać wywoływując narzędzie z opcją `--help`. Eksperymenty wykazały, że domyślne wartości w większości przypadków są odpowiednie (nie licząc argumentu `--outmodel`, który ma inny charakter). Podamy teraz ich dokładniejszy opis:

- `-d --datainmemory` – Włącza przechowywanie całego zbioru danych (zbioru treningowego i zbioru do ewaluacji) w pamięci ulotenej komputera. Proces trenowania jest wtedy szybszy, ale za to zwiększają się też wymagania pamięciowe.
- `-i --iternum` – Liczba iteracji, czyli przejść po całym zbiorze treningowym, po której trenowanie zostanie zatrzymane. Eksperymenty pokazały, że optymalna wartość tego parametru znajduje się w przedziale $[10, 20]$.
- `-b --batchsize` – Rozmiar „porcji" zbioru treningowego wykorzystywany w algorytmie SGD (zob. rozdział 6.2). Z literatury wynika, że wartości z przedziału $[15, 100]$ dają najlepsze rezultaty.
- `-r --regvar` – Stała opdowiadająca za kontrolę absolutnych wartości parametrów modelu opisana w rozdziale 6.2.
- `-s --scale0` – Stała opisana w rozdziale 6.2. Domyślnie równa 1, co daje szybką zbierzność algorytmu (w literaturze zwykle podawane są mniejsze wartości, np. 0.1, ale w przypadku trenowania tagera wartość 1 daje bardzo dobre rezultaty).
- `-t --tau` – Liczba przejść po zbiorze treningowym, po której „skala" (początkowo równa `scale0`) zmniejszy się dwukrotnie. Opis stałej można znaleźć w rozdziale 6.2.
- `-w --workersnum` – Liczba wątków oddelegowanych do obliczania gradientu. Ustawienie tej wartości np. na 4 powinno przynieść wyraźne przyspieszenie algorytmu trenowania, jednak testy, czy jej dalsze zwiększanie przyniesie dalsze przyspieszenie (przy założeniu, że jest dostępna odpowiednia liczba rdzeni) nie zostały przeprowadzone. Stała ma wpływ na działanie programu tylko wtedy, gdy tager zostanie uruchomiony w trybie współbieżnym.
- `-o --outmodel` – Ścieżka do pliku, w którym ma zostać zapisany wytrenowany model. Jeśli nie zostanie podana, model nie zostanie zapisany na dysku.

9.6 Walidacja krzyżowa

Uruchamienie programu w trybie walidacji krzyżowej jest bardzo podobne jak w trybie trenowania. Program przyjmuje w zasadzie takie same opcje i dodatkową opcję `-k --kcrossval`, która steruje liczbą części na które zostanie podzielony zbiór danych w trakcie walidacji krzyżowej. Aby zobaczyć opje programu w tym trybie, można uruchomić tager następującą komendą:

```
./tager cv --help
```

10 Format LINC

Format LINC stanowi wewnętrzny, domyślny format na którym pracuje tager. Narzędzie w trybie tagowania można również uruchomić na pliku wejściowym w formacie Morphosa, ale w trybie trenowania i walidacji krzyżowej już tylko na zbiorach danych w formacie LINC.

Plik w formacie LINC składa się z listy opisów zdań, a opis zdania – z listy opisów słów. Opis zdania rozpoczyna się od słowa kluczowego `Sent`, po którym następuję lista opisów słów. Opis słowa rozpoczyna się od jego formy ortograficznej w tekście zapisanej w cudzysłowie. Jeśli forma zawiera znak cudzysłowia, wtedy każde jego wystąpienie jest powtórzone (czyli np. pojedynczy znak " zapisany w cudzysłowie porzyjmie postać """"). Po formie ortograficznej następują, w kolejnych liniach, poszczególne znaczniki reprezentujące interpretacje formy. Każdy znacznik poprzedzony jest:

- Symbolem - lub *, gdzie drugi z nich oznacza że znacznik został wybrany w procesie ujednoznaczniania. Dopuszczalna jest dowolna liczba wybranych znaczników większa od 0.
- Prawdopodobieństwem, które zostało mu przypisane na etapie ujednoznaczniania. Powinny sumować się do 1, ale na wszelki są w ramach parsera plików LINC ponownie normalizowane.

Dla pojedynczego słowa nie należy mieszać powyższych reprezentacji (chociaż z technicznego punktu widzenia jest to możliwe).

Przykładowy fragment pliku w formacie LINC:

```
Sent
 "Zatrzasnął"
  * v.C3sopod
  - v.C3srpod
  - v.C3szpod
 "drzwi"
  * n.App2
  - n.Gpp2
  - n.Npp2
  - n.Vpp2
 "od"
  - n.Gpf
  * prep.
 "mieszkania"
  - n.Apn
  * n.Gsn
  - n.Npn
  - n.Vpn
  - v.NGsnn
 "." Nps
  * interp.
Sent
```

```
 "Bohaterem"
  * n.Iso
 "powieści"
  - n.Apf
  - n.Dsf
  - n.Gpf
  * n.Gsf
  - n.Lsf
  - n.Npf
  - n.Vpf
  - n.Vsf
 "Październiewskiego"
  - m.
  * n.Gso
...
```