Uniwersytet imienia Adama Mickiewicza
w Poznaniu
Wydział Matematyki i Informatyki

Praca doktorska

„Algorytmy dopasowania wyrazów metodami statystycznymi z wykorzystaniem wielowątkowości i symetryzacji obliczeń”

mgr Arkadiusz Szał

Kierownik pracy:
Prof. dr hab. Krzysztof Jassem
Pracownia Systemów Informacyjnych

Poznań 2013

Dziękuję Panu Profesorowi
Krzysztofowi Jassem
za poświęcony czas,
a także Panu Doktorowi
Marcinowi Junczysowi-Dowmuntowi
za pomoc w rozwiązaniu
wielu problemów.

Spis treści

Oświadczenie

Ja niżej podpisany Arkadiusz Szał, doktorant Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu, oświadczam, że przedkładaną pracę pod tytułem „Algorytmy automatycznego dopasowania wyrazów metodami statystycznymi z wykorzystaniem wielowątkowości i symetryzacji obliczeń” napisałem samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania rozprawy lub jej istotnych części innym osobom, ani nie odpisywałem tej rozprawy lub jej istotnych części od innych osób.

Równocześnie wyrażam zgodę na to, że gdyby powyższe oświadczenie okazało się nieprawdziwe, decyzja o wydaniu mi dyplomu zostanie cofnięta.

Arkadiusz Szał

Wstęp

W pracy „Algorytmy automatycznego dopasowywania wyrazów metodami statystycznymi z wykorzystaniem wielowątkowości i symetryzacji obliczeń” przedstawiam nowy sposób podejścia do symetryzacji dopasowań wyrazów. Dotychczasowe badania uwzględniały symetryzację na końcu procesu automatycznego dopasowywania wyrazów, w moich badaniach postanowiłem zastosować ten mechanizm wewnątrz poszczególnych etapów przetwarzania odpowiadającym poszczególnym modelom statystycznym. Poprzez takie podejście do tematu uzyskałem więcej informacji o możliwych dopasowaniach pomiędzy wyrazami w poszczególnych modelach statystycznych. To z kolei pozwoliło z większą precyzją wyszukać najbardziej prawdopodobne dopasowania. Poprawa jakości dopasowania wyrazów przekłada się w konsekwencji na wyższą jakość systemów tłumaczenia statystycznego, korzystających z wyników procesu dopasowywania wyrazów w dwujęzycznych korpusach trenujących.

W rozdziale 1. omawiam istotę automatycznego dopasowywania wyrazów, opisując szczegółowo poszczególne modele statystyczne stosowane w tym procesie. Modele statystyczne pozwalają wyznaczyć prawdopodobieństwo dopasowania pomiędzy poszczególnymi wyrazami w korpusie dwujęzycznym. Ponadto przedstawiam prace poświęcone dopasowywaniu wyrazów, które miały wpływ na rozwój tej dziedziny nauki. W rozdziale tym opisuję również dostępne narzędzia dopasowywania wyrazów: Giza++, MGiza++ i PGiza++ wraz z ich algorytmami działania. Aplikacje te zwracają jednakowe wyniki dopasowań wyrazów, gdyż korzystają z tych samych modeli statystycznych, a różnica w ich działaniu polega na różnym dysponowaniu dostępnymi zasobami procesora. Giza++ wykonuje obliczenia szeregowo, podczas gdy MGiza++ wykorzystuje wielowątkowość, pozwalającą na wykonywanie części obliczeń równolegle. Z kolei PGiza++ pozwala na wykonywanie obliczeń na wielu maszynach połączonych w jedną sieć, tworzącą klaster obliczeniowy. Przedstawiam także narzędzia pozwalające zintegrować cały proces dopasowywania wyrazów. Są to takie rozwiązania jak LoonyBin, czy szeroko stosowany Moses. Narzędzia te pozwalają zintegrować zarówno aplikacje dopasowywania wyrazów, jak i aplikacje wykonujące symetryzację końcową procesu dopasowywania.

W rozdziale 2. przedstawiam rodzaje symetryzacji dopasowań wyrazów wraz z ich zaletami i wadami. Opisuję również możliwe etapy przetwarzania, w których można zastosować mechanizm symetryzacji:

- wewnątrz procesu dopasowywania wyrazów, po każdej iteracji kolejnych modeli statystycznych,
- na plikach wynikowych procesu dopasowywania wyrazów.

Przedstawiam też miary określające trafność dopasowania (współczynnik AER) i jakość dopasowania (miara BLEU). Miary te są wykorzystywane w rozdziale 4. do porównania otrzymanych wyników obliczeń z wynikami dostępnymi w opublikowanej literaturze.

W rozdziale 3. pracy opisuję kluczowy wynik badań, czyli nowy algorytm symetryzacji dwukierunkowych dopasowań wyrazów metodami statystycznymi. W tym podejściu symetryzacja jest wykonywana po każdym kroku iteracji modelu statystycznego. W przeciwieństwie do dostępnych publikacji, gdzie symetryzacja jest wykonywana na plikach wynikowych procesu dopasowywania wyrazów.

Przedstawiam między innymi zmiany, jakie wprowadziłem w poszczególnych modelach statystycznych, aby wykorzystać mechanizm symetryzacji, metody przyśpieszenia obliczeń poprzez zastosowanie wielowątkowości obliczeń, a także złożoność obliczeniową całego procesu. Efektem tych prac jest narzędzie SyMGiza++, w którym zaimplementowano omówione mechanizmy poprawy jakości i szybkości procesu dopasowywania wyrazów.

Wyniki dotyczące szybkości działania dostępnych programów dopasowania wyrazów przedstawiam w rozdziale 4. Rozdział ten zawiera także omówienie wpływu symetryzacji na poprawę trafności dopasowań wyrazów i jakości tłumaczenia, przedstawione w postaci odpowiednich tabel porównujących współczynnik AER i miarę BLEU narzędzi Giza++ i SyMGiza++. Porównania dokonano na podstawie dostępnych publikacji naukowych, biorąc za dane wejściowe dokładnie te same korpusy dwujęzyczne.

Pracę zamyka rozdział 5., stanowiący podsumowanie otrzymanych wyników wraz z zarysem możliwych kroków dalszego rozwoju zaproponowanych metod poprzez optymalizację i modyfikację algorytmów symetryzacji, a także zastosowanie innych metod symetryzacji wewnątrz procesu obliczeń. Takie działania powinny przyczynić się do dalszej poprawy trafności dopasowań, a co za tym idzie i jakości tłumaczenia.

Ponadto, w pracy zawarłem dodatki ukazujące szczegółowe algorytmy symetryzacji poszczególnych modeli statystycznych, a także przedstawiające proces instalacji i konfiguracji zaimplementowanego narzędzia SyMGiza++. Narzędzie to zostało udostępnione

publicznie na licencji GNU (General Public License) na stronie http://psi.amu.edu.pl wraz ze szczegółowym opisem instalacji i konfiguracji poszczególnych elementów.

Rozdział 1

Wprowadzenie

1.1. Idea tłumaczenia statystycznego

Translatory automatyczne zyskują coraz większe uznanie na świecie. Przyczyniają się do tego takie rozwiązania jak Google Translate czy Microsoft Bing Translator. Istniejące rozwiązania korzystają z różnych technik tłumaczenia. Tłumaczenie oparte na regułach polega na zastosowaniu dla tłumaczonego zdania odpowiedniego zestawu reguł wybranego ze zbioru uprzednio opracowanego manualnie lub automatycznie. Tłumaczenie przez analogię opiera się na przyrównaniu danego zdania (wyrażenia) do zdania występującego w zasobie systemu. Na tej podstawie tworzy się tłumaczenie wynikowe. Tłumaczenie statystyczne korzysta z bazy dopasowań wyrazów w dokumentach otrzymanych za pomocą odpowiednich metod statystycznych. Analizując dwujęzyczne korpusy zdań, w których każde zdanie ma swój odpowiednik w drugim języku, system tworzy możliwe dopasowania słów, opierając się na modelach dopasowań. Im więcej dostępnych danych wejściowych składających się z korpusów dwujęzycznych, tym lepsza i bardziej obszerna jest baza powiązań, a co za tym idzie, wyższa jest jakość tłumaczenia końcowego.

Praca ta przedstawia statystyczne metody dopasowania wyrazów z użyciem nowych technik polepszających jakość i szybkość obliczeń.

Technika dopasowywania wyrazów sprowadza się do dwóch kroków:

- utworzenia pokaźnej bazy korpusów dwujęzycznych (każde zdanie ma swój odpowiednik w drugim języku),
- dopasowania wyrazów z wykorzystaniem dostępnych metod statystycznych.

1.2. Dopasowywanie wyrazów metodami statystycznymi

Definicja 1. Dopasowywanie korpusów (ang. corpus alignment) – etap przygotowania plików wejściowych procesu tłumaczenia statystycznego, polegający na powiązaniu odpowiadających sobie par zdań, będących swoim wzajemnym tłumaczeniem, w jeden korpus, zawierający listę zdań w języku źródłowym z ich odpowiednikami w języku docelowym.

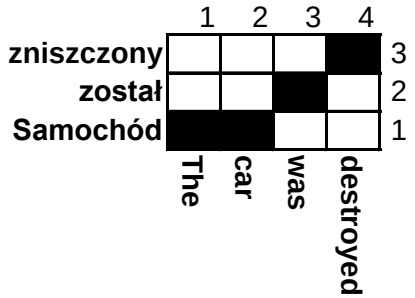
Definicja 2. Dopasowywanie wyrazów (ang. word alignment) – etap tłumaczenia statystycznego, polegający na powiązaniu odpowiadających sobie wyrazów między parą zdań, które stanowią wzajemne tłumaczenia.

Definicja 3. Dopasowywanie skierowane (ang. directed alignment) – etap tłumaczenia statystycznego, polegający na powiązaniu odpowiadających sobie wyrazów w jednym kierunku procesu dopasowywania wyrazów. Zdanie docelowe (wyjściowe) tego procesu jest tłumaczeniem zdania źródłowego (wejściowego). W ten sposób otrzymujemy tłumaczenia 1 do n (jeden wyraz ze zdania źródłowego ma n (0, 1 lub więcej) powiązań z wyrazami ze zdania docelowego).

Wynikiem procesu dopasowywania jest **macierz dopasowania** wyrazów o wymiarze m x n, gdzie m i n oznaczają liczbę wyrazów w zdaniach będących swoimi wzajemnym tłumaczeniami.

Przykład 1. Na rysunku 1. przedstawiona jest macierz dopasowania dla pary zdań:

- polskiego: *Samochód został zniszczony.*
- angielskiego: *The car was destroyed.*



Rysunek 1. Macierz dopasowania wyrazów.

Dopasowywanie wyrazów jest istotnym etapem procesu **statystycznego tłumaczenia automatycznego**. W metodzie tej zbierany jest zestaw dokumentów i ich tłumaczeń zwany **korpusem dwujęzycznym.** W korpusie dwujęzycznym wyznacza się pary zdań będących swoimi odpowiednikami – proces ten nazywany jest **dopasowywaniem korpusów.** Dla każdej pary zdań tworzy się macierz dopasowania wyrazów. Na bazie powstałych macierzy dopasowań system tłumaczy zadany tekst źródłowy. Im lepsza baza powiązań, tym tłumaczenie jest wyższej jakości.

Przykład 2. Niniejszy przykład obrazuje przykładowe działanie algorytmu statystycznego tłumaczenia automatycznego na podstawie niewielkiego korpusu dwujęzycznego.

Załóżmy, że dany jest następujący korpus polsko-angielski (zaczerpnięty z [1]).

Działania podjęte w wyniku rezolucji Parlamentu: Patrz protokół
Składanie dokumentów: patrz protokół
Oświadczenia pisemne (art. 116 Regulaminu): patrz protokół
Action taken on Parliament's resolutions: see Minutes
Documents received: see Minutes
Written statements (Rule 116): see Minutes

Tabela 1. Przykładowy korpus dwujęzyczny

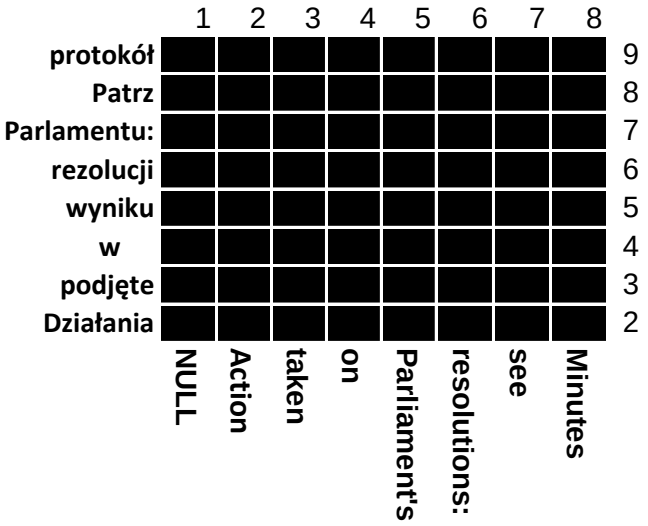
W tabeli 1. każdemu zdaniu w języku polskim odpowiada jego tłumaczenie w języku angielskim – wyznaczenie mapowania pomiędzy odpowiadającymi sobie zdaniami dokonywane jest w fazie dopasowywania korpusów. Następnie każdemu wyrazowi korpusu jest przypisywany indywidualny identyfikator wraz z krotnością występowania danego wyrazu w korpusie. Efekt tej operacji obrazuje tabela 2.

2 Działania 1
3 podjęte 1
4 w 1
5 wyniku 1
6 rezolucji 1
7 Parlamentu 1
8 Patrz 1
9 protokół 3
10 Składanie 1
11 dokumentów: 1
12 patrz 2
13 Oświadczenia 1
14 pisemne 1
15 (art. 1
16 116 1
17 Regulaminu): 1

Tabela 2. Wyrazy z korpusu dwujęzycznego i ich identyfikatory. Pierwsza liczba oznacza identyfikator wyrazu, natomiast ostatnia liczba określa ilość wystąpień danego wyrazu w korpusie.

W początkowym etapie fazy dopasowywania wyrazów, każdy wyraz ze zdania korpusu źródłowego zostaje dopasowany do każdego wyrazu odpowiadającego mu zdania korpusu

docelowego. Zakłada się przy tym, że wszystkie mapowania wyrazów dla danej pary zdań są jednakowo prawdopodobne. Sytuację tę obrazuje rysunek 2.



Rysunek 2. Macierz dopasowania wyrazów: „każdy z każdym" dla pierwszej pary zdań.

Na podstawie danych zebranych ze wszystkich zdań korpusu oblicza się prawdopodobieństwo dopasowania pomiędzy poszczególnymi wyrazami korpusu (zwane prawdopodobieństwem tłumaczenia).

6 2 0.125
6 3 0.125
6 4 0.125
6 5 0.125
6 6 0.125
6 7 0.125
6 8 0.125
6 9 0.125
7 2 0.0446429
7 3 0.0446429
7 4 0.0446429
7 5 0.0446429
7 6 0.0446429
7 7 0.0446429
7 8 0.0446429
7 9 0.167092
7 10 0.0714286
7 11 0.0714286
7 12 0.122449
7 13 0.0510204
7 14 0.0510204
7 15 0.0510204
7 16 0.0510204
7 17 0.0510204
8 2 0.0446429
8 3 0.0446429
8 4 0.0446429
8 5 0.0446429
8 6 0.0446429
8 7 0.0446429
8 8 0.0446429
8 9 0.167092
8 10 0.0714286
8 11 0.0714286
8 12 0.122449
8 13 0.0510204
8 14 0.0510204
8 15 0.0510204
8 16 0.0510204
8 17 0.0510204

Tabela 3. Fragment dopasowania wyrazów: „każdy z każdym". Kolumna pierwsza i druga oznaczają identyfikatory wyrazu źródłowego (angielskiego) i docelowego (polskiego), natomiast kolumna trzecia określa prawdopodobieństwo tłumaczenia wyrazów.

Tabela 3. przedstawia fragment pliku zawierającego prawdopodobieństwo tłumaczenia wyrazów. Pierwsze dwie kolumny oznaczają identyfikatory wyrazu angielskiego i polskiego.

Trzecia kolumna zawiera prawdopodobieństwo tłumaczenia obliczone w wyniku jednej iteracji modelu 1. W tym przypadku przedstawiono prawdopodobieństwa dla wyrazów angielskich o identyfikatorach 6, 7 i 8. Można zauważyć, że suma prawdopodobieństw dla poszczególnych wyrazów angielskich wynosi 1. Są one jednakowe, gdyż w fazie inicjalizacji modelu 1. prawdopodobieństwa te dla każdego powiązania wyrazów przyjmują taką samą wartość. Szczegóły obliczeń dla poszczególnych modeli zostały przedstawione w rozdziale 1.4.

Dla tak wyznaczonych danych początkowych stosuje się iteracyjnie bardziej zaawansowane modele statystyczne powiązań wyrazów, biorące pod uwagę m.in. kolejność występowania wyrazów w zdaniu (patrz rozdział 1.4).

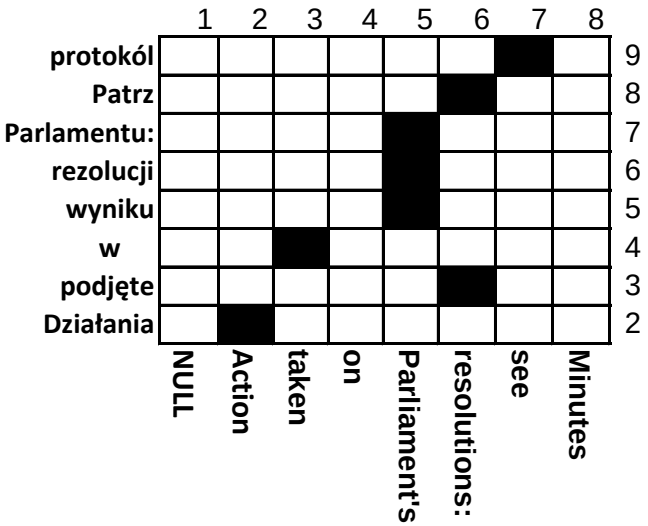
Wykonanie pięciu iteracji modelu 1., 2., 3., 4. i modelu Markowa dla przykładowego korpusu zwróci zestaw danych, którego fragment zaprezentowano w tabeli 4.

```
2 2 1
3 4 1
4 2 0.125
4 3 0.125
4 4 0.125
4 5 0.125
4 6 0.125
4 7 0.125
4 8 0.125
4 9 0.125
5 5 0.333333
5 6 0.333333
5 7 0.333333
6 3 0.5
6 8 0.5
7 9 1
```

Tabela 4. Fragment pliku prezentującego powiązania wyrazów w korpusie polsko-angielskim po zastosowaniu pięciu iteracji modelu 1., 2., 3., 4., 5. i modelu Markowa.

Można ponownie zauważyć, że prawdopodobieństwo tłumaczenia wszystkich wyrazów drugiego języka z korpusu dla konkretnego wyrazu zawsze sumuje się do 1.

Na podstawie powyższego pliku tworzy się macierz dopasowania wyrazów przedstawioną na rysunku 3. Dla każdego wyrazu docelowego (kolumna 2) szuka się powiązania o największym prawdopodobieństwie (kolumna 3). Na tej podstawie można wyznaczyć powiązanie z wyrazem źródłowym (kolumna 1).



Rysunek 3. Macierz dopasowania wyrazów wygenerowana z pliku powiązań wyrazów w korpusie polsko-angielskim dla pierwszej pary zdań.

Reprezentacja plikowa wygenerowana przez system ma postać przedstawioną w tabeli 5.

 # Sentence pair (1) source length 7 target length 8 alignment score : 0.000357497
Działania podjęte w wyniku rezolucji Parlamentu: Patrz protokół
NULL ({ }) Action ({ 1 }) taken ({ 3 }) on ({ }) Parliament's ({ 4 5 6 }) resolutions: ({ 2 7 }) see ({ 8 }) Minutes ({ })
 # Sentence pair (2) source length 4 target length 4 alignment score : 0.22522
Składanie dokumentów: patrz protokół
NULL ({ 3 }) Documents ({ 1 }) received: ({ 2 }) see ({ 4 }) Minutes ({ })
 # Sentence pair (3) source length 6 target length 7 alignment score : 0.000224203
Oświadczenia pisemne (art. 116 Regulaminu): patrz protokół
NULL ({ 6 }) Written ({ }) statements ({ }) (Rule ({ 5 }) 116): ({ 1 2 3 4 }) see ({ 7 }) Minutes ({ })

Tabela 5. Reprezentacja plikowa macierzy dopasowania dla korpusu polsko-angielskiego w postaci pliku.

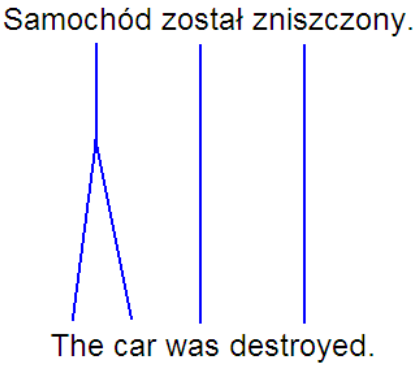
Cyfry w nawiasach oznaczają identyfikator wyrazu ze zdania docelowego, który został przyporządkowany danemu wyrazowi. Wyraz pusty (NULL) nie ma w tym przypadku identyfikatora, więc przyporządkowanie identyfikatorów do wyrazów rozpoczyna cyfra 1. Na podstawie tak utworzonej macierzy dopasowań system automatycznego tłumaczenia przypasowuje do danego wyrazu jego odpowiednik. Jako przykład niech posłuży nam zdanie: „*Z dokumentów składano protokół”*, którego tłumaczenie na podstawie powyższej macierzy tłumaczeń będzie miało postać: „*? received: Documents see”*. Znak zapytania zostaje wstawiony, gdy dla danego wyrazu nie istnieje jego odpowiednik w macierzy dopasowań. Jak można zauważyć otrzymane tłumaczenie dobre jest dalekie od oczekiwań. Wynika to z bardzo małego rozmiaru korpusu, dla których zastosowano modele statystyczne.

Typy dopasowań

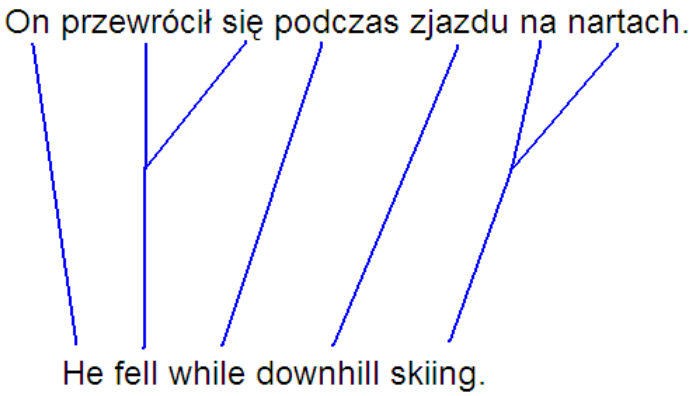
Wyraz w języku źródłowym może mieć tłumaczenie składające się z kilku wyrazów w języku docelowym i na odwrót: kilku wyrazom w języku źródłowym może odpowiadać jeden wyraz w języku docelowym. W związku z tym wyróżniamy trzy podstawowe sposoby powiązań wyrazów będących swoim tłumaczeniem:

1) Dopasowanie 1 do n (n> = 1),
2) Dopasowanie n do 1 (n >=1),
3) Dopasowanie n do m (n, m > 1).

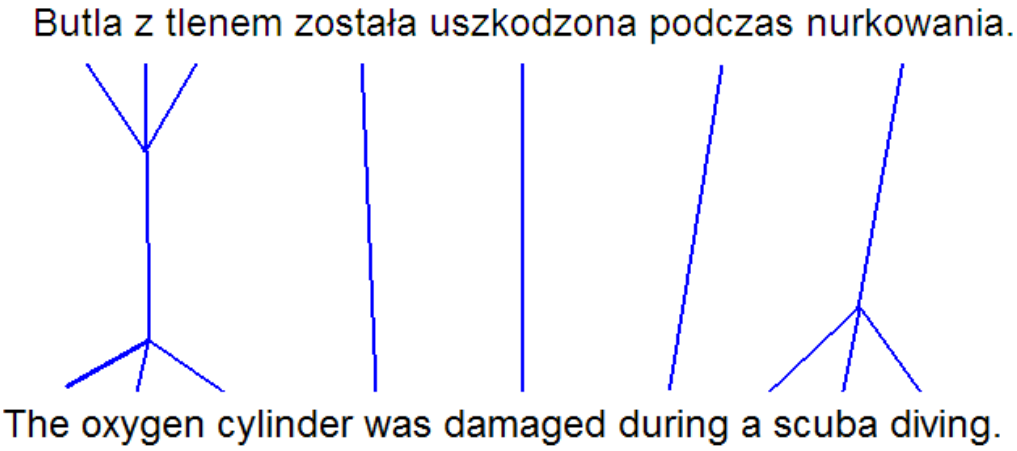
Przykłady powyższych typów powiązań obrazują odpowiednio: rysunek 4., rysunek 5. i rysunek 6.



Rysunek 4. Dopasowanie 1 do n , gdzie n>=1 (jeden wyraz polski jest powiązany z n wyrazami angielskimi).



Rysunek 5. Dopasowanie n do 1, gdzie n>=0 (jeden wyraz angielski jest powiązany z n wyrazami polskimi).



Rysunek 6. Dopasowanie n do m, gdzie n i m >=0 (n wyrazów angielskich jest powiązanych z m wyrazów polskich).

1.3. Zarys historyczny – przegląd prac poświęconych dopasowywaniu wyrazów

Pierwszą pracą sugerującą zastosowanie mechanizmów statystyki i kryptoanalizy do tłumaczenia z jednego języka naturalnego do innego języka była praca Warren'a Weaver'a [2] z roku 1949, a opublikowana w 1955. Była to głównie praca teoretyczna, ze względu na tamtejsze ograniczenia technologiczne. Potrzebne było kilka dziesięcioleci, aby tę tematykę rozwinąć i zastosować w praktyce.

W latach dziewięćdziesiątych dwudziestego wieku wzrosło zainteresowanie automatycznym dopasowywaniem wyrazów metodami statystycznymi. W pierwszych pracach poświęconych tej tematyce [3], [4] autorzy proponowali użycie metod statystycznych do tłumaczenia automatycznego z języka francuskiego na angielski. Algorytm opierał się na wyliczeniu prawdopodobieństw tłumaczeń danego wyrazu angielskiego na wyraz francuski. Pokazano, że tak obliczone prawdopodobieństwo może być użyte w modelu statystycznym procesu tłumaczenia całych zdań.

W kolejnych latach powstało kilka prac ukazujących możliwość dopasowywania wyrazów bez informacji o ich znaczeniu [5], [6], [7], [8]. Praca Brown'a Lai'a i Mercer'a (1991) zawierała algorytm dopasowywania wyrazów na podstawie informacji o ilości słów w zdaniu, natomiast praca Gale'a i Church'a (1991) opierała się na algorytmie zliczającym ilość znaków w zdaniu.

Przełom nastąpił w 1993 roku, kiedy to grupa badaczy w IBM ponownie przyjrzała się zastosowaniu statystyki do tłumaczenia automatycznego. Wyniki ich badań zostały opisane

w pracy [9], która stała się jedną z najważniejszych publikacji w dziedzinie tłumaczenia automatycznego. Brown wraz z badaczami zaproponowali modele statystyczne, których zastosowanie znacznie poprawia wyniki dopasowań wyrazów. Szczegóły dotyczące poszczególnych modeli statystycznych zostały przedstawione w rozdziale 1.4.

1.4. Opis modeli dopasowania (modele 1. – 6., HMM)

W zastosowanych technikach dopasowywania wyrazów metodami statystycznymi wykorzystuje się modele IBM [10] opracowane w 1993 roku przez Brown'a, Pietra i Mercer'a [9]. Ponadto opracowano dwa rozszerzenia – model Markowa HMM [11] i model 6. [12]. W pracy szczegółowo opiszę aparat matematyczny wykorzystywany w poszczególnych modelach. Kolejne modele statystyczne stają się bardziej skomplikowane pod względem zastosowanych technik dopasowań, poprzez wykorzystanie szeregu zależności między wyrazami w zdaniach, będących wzajemnymi tłumaczeniami. W każdym modelu powstają specyficzne macierze prawdopodobieństwa dopasowania wyrazów, prawdopodobieństwa zmiany pozycji danego wyrazu w zdaniu będącym tłumaczeniem i wiele innych parametrów pomocniczych.

Algorytm wyznaczania parametrów modelu zwany algorytmem EM (ang. Expectation-Maximilization) [13] składa się z dwóch faz. W fazie pierwszej zwanej E-step wykorzystuje się parametry z poprzedniego modelu (lub iteracji). Na ich podstawie tworzy się parametry do wykorzystania w aktualnej iteracji. Jest to krok obliczania oczekiwanych wartości wiarygodności (jakby parametry ukryte zostały zaobserwowane). W kroku drugim – M-step wygenerowane parametry są wykorzystane do wyliczenia prawdopodobieństwa w kolejnym modelu. Jest to krok maksymalizacji, który oblicza oszacowania największej wiarygodności parametrów poprzez maksymalizację oczekiwanych wiarygodności kroku E. Proces ten jest kontynuowany aż do uzyskania zbieżności. Algorytm EM jest wykorzystywany do wyznaczania wartości największej wiarygodności modeli probabilistycznych, gdyż model zależy od ukrytych parametrów.

Algorytm EM wykorzystuje się w grafice komputerowej, do klastrowania danych w uczeniu maszynowym, przy automatycznej indukcji probalistycznych gramatyk bezkontekstowych, a także w modelach Markowa.

Proces dopasowywania wyrazów za pomocą statystycznych modeli dopasowania składa się z kilku iteracji następujących modeli:

- model 1.: 5 iteracji tego modelu jest z reguły wystarczającą ilością, aby uzyskać wysoką trafność dopasowań,
- model 2., zastępowany często przez model HMM, który daje lepszą trafność dopasowania wyrazów,
- model 3. i 4.

Dodatkowo można uruchomić kilka iteracji modelu 5., a także modelu 6., który jest kombinacją modelu HMM i modelu 4.

Poniżej zaprezentowano opis matematyczny poszczególnych modeli wraz z ich zastosowaniem w dopasowywaniu wyrazów w małym korpusie dwujęzycznym.

Definicja 4. Zbieżność monotoniczna – mówimy, że ciąg liczb rzeczywistych $(a_n)_{n\in N}$ jest monotonicznie zbieżny do liczby a, jeśli $(a_n)_{n\in N}$ jest ciągiem monotonicznym zbieżnym do liczby a.

Twierdzenie 1. Zbieżność ciągu monotonicznego - ciąg monotoniczny jest zbieżny wtedy i tylko wtedy gdy jest ograniczony.

Niech X będzie dowolnym zbiorem oraz $f_n, f: X \to R$. Mówimy, że ciąg $(f_n)_{n\in N}$ jest zbieżny monotonicznie do funkcji f jeśli:

- $(\forall n \in N)(\forall x \in X)(f_n(x) \le f_{n+1}(x))$ lub $(\forall n \in N)(\forall x \in X)(f_n(x) \ge f_{n+1}(x))$ oraz
- $(f_n)_{n\in N}$ jest zbieżny punktowo do funkcji f (tzn. dla każdego $x \in X$: $f(x) = \lim_{n\to\infty} f_n(x)$)
- $(f_n)_{n\in N}$ jest zawsze niemalejący lub zawsze nierosnący.

Twierdzenie 2. Twierdzenie Bayesa – Niech X będzie pewnym zdarzeniem, T zaś pewną teorią. $P(X)$ jest obserwowanym prawdopodobieństwem X, zaś $P(X|T)$ to prawdopodobieństwo, że X nastąpi według teorii T. Z kolei $P(T)$ to prawdopodobieństwo, że teoria T jest prawdziwa, $P(T|X)$ to prawdopodobieństwo, że teoria T jest prawdziwa, jeśli zaobserwowano X.

Twierdzenie Bayesa wykorzystuje się w modelach statystycznych dopasowania wyrazów przy obliczaniu prawdopodobieństwa warunkowego powiązania dwóch zdań e i f.

W modelu 1. otrzymujemy wzór:

$$\Pr(f \mid e) = \frac{\Pr(f)\Pr(e \mid f)}{\Pr(e)},$$

który jest interpretowany jako prawdopodobieństwo warunkowe, że zdanie f jest tłumaczeniem zdania e. Prawdopodobieństwo to powstaje jako iloczyn prawdopodobieństwa wystąpienia zdania f i prawdopodobieństwa wystąpienia zdania e z jego tłumaczeniem f, podzielone przez prawdopodobieństwo wystąpienia zdania e.

Model 1

Model 1. został dokładnie opisany w pracy [9]. W początkowej fazie algorytmu każdy wyraz ze zdania wejściowego jest łączony z każdym wyrazem ze zdania wyjściowego. Proces ten można opisać równaniem [14]:

$$\hat{\mathbf{e}} = \arg\max_{e} P(\mathbf{E} = \mathbf{e} \mid \mathbf{F} = \mathbf{f}), \tag{1}$$

gdzie $\mathbf{E}$ i $\mathbf{F}$ są zmiennymi losowymi przebiegającymi odpowiednio po wszystkich zdaniach $\mathbf{e}$ korpusu źródłowego i zdaniach $\mathbf{f}$ korpusu docelowego. Dla ustalonego zdania $\mathbf{f}$ zdanie $\hat{\mathbf{e}}$ maksymalizuje powyższą funkcję prawdopodobieństwa i jest najbardziej prawdopodobnym tłumaczeniem $\mathbf{f}$. Zastosowanie warunkowego prawdopodobieństwa sprawia, że proces tłumaczenia jest procesem ukierunkowanym, który tworzy zdanie docelowe po zaobserwowaniu zdania źródłowego.

Po przekształceniu równania 1. za pomocą twierdzenia Bayesa (twierdzenie 2.):

$$P(\mathbf{E} = \mathbf{e} \mid \mathbf{f} = \mathbf{f}) = \frac{P(\mathbf{F} = \mathbf{f} \mid \mathbf{E} = \mathbf{e})P(\mathbf{E} = \mathbf{e})}{P(\mathbf{f} = \mathbf{f})}, \tag{2}$$

otrzymujemy:

$$\hat{\mathbf{e}} = \arg\max_{e} P(\mathbf{E} = \mathbf{e} \mid \mathbf{F} = \mathbf{f})P(\mathbf{E} = \mathbf{e}). \tag{3}$$

Otrzymane równanie (3) jest podstawowym równaniem tłumaczenia statystycznego. Komponent $P(\mathbf{F} = \mathbf{f} \mid \mathbf{E} = \mathbf{e})$ nazywamy modelem tłumaczenia, natomiast $P(\mathbf{E} = \mathbf{e})$ - modelem języka.

W modelu 1., po uwzględnieniu zależności między wyrazami otrzymujemy:

$$\Pr(\mathbf{f} \mid \mathbf{e}) = \frac{\varepsilon(m \mid l)}{(l+1)^m} \sum_a \prod_{j=1}^{m} t(f_j \mid e_{a_j}), \tag{4}$$

gdzie l i m są długościami zdania wejściowego i wyjściowego, a jest dopasowaniem skierowanym między zdaniami, t jest skierowanym prawdopodobieństwem tłumaczenia pomiędzy wyrazem wejściowym i wyjściowym, a ε stałą zależną od długości poszczególnych zdań, e i f oznaczają pojedyncze wyrazy. Równanie to interpretujemy jako prawdopodobieństwo, że maszyna tłumacząca dla zdania **e** zwróci zdanie **f** jako jego tłumaczenie.

Prawdopodobieństwo tłumaczenia poszczególnych wyrazów e i f jest opisane wzorem:

$$t(f \mid e) = \frac{\sum_{S=1}^{S} c(f \mid e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}{\sum_{f'} \sum_{S=1}^{S} c(\dot{f} \mid e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}, \tag{5}$$

gdzie S jest liczbą zdań w korpusie, $c(f \mid e; \mathbf{f}, \mathbf{e})$ jest częstością powiązań słowa wejściowego i wyjściowego w wyrazach ze zdania **f** i **e**. Częstość powiązań $c(f \mid e; \mathbf{f}, \mathbf{e})$ jest obliczana z wartości t za pomocą dwóch równań:

$$c(f \mid e; \mathbf{f}, \mathbf{e}) = \sum_a \Pr(a \mid \mathbf{f}, \mathbf{e}) \sum_{i,j} \delta(f, f_j) \delta(e, e_i), \tag{6}$$

$$\Pr(a \mid \mathbf{f}, \mathbf{e}) = \frac{\prod_{j=1}^{m} t(f_j \mid e_{a_j})}{\sum_a \prod_{j=1}^{m} t(f_j \mid e_{a_j})}, \tag{7}$$

gdzie δ jest deltą Kroneckera:

$$\delta(i, j) = \begin{cases} 1 & \text{jezeli i = j} \\ 0 & \text{w przeciwnym wypadku} \end{cases} \tag{8}$$

Suma po wszystkich prawdopodobieństwach t danego wyrazu e sumuje się do 1:

$$\sum_f t(f \mid e) = 1. \tag{9}$$

Skoro suma wszystkich prawdopodobieństw tłumaczenia t danego wyrazu e sumuje się do jedynki, a celem dopasowania wyrazów jest nadanie jak największego

prawdopodobieństwa t wybranej parze wyrazów, możemy stwierdzić, że taka para w ogólności jest zbieżna monotonicznie (twierdzenie 2.) do 1.

Jako przykład opisujący działanie modelu 1. zostanie wykorzystany fragment korpusu polsko-angielskiego. W tym celu wprowadzimy następujące definicje:

Definicja 5. Normalizacja tekstu – proces przetwarzania tekstu, nadający mu spójną formę, ułatwiającą dalszą interpretację. Często stosowana jako etap wstępny procesu dopasowywania wyrazów. W normalizacji występują następujące procesy:

- zmiana wielkości liter (na małe lub duże),
- ujednolicenie wyrażeń numerycznych,
- usunięcie znaków specjalnych,
- ujednolicenie skrótów,
- usunięcie znaków interpunkcyjnych,
- usunięcie (lub zmiana) znaków diaktrycznych.

Definicja 6. Tokenizacja tekstu – proces analizy tekstu polegający na grupowaniu ciągów znaków w tokeny. Każdy token jest ograniczony znakami zdefiniowanymi jako separatory (np. spacja). Tokeny to głównie wyrazy języka naturalnego, liczby, nazwy własne lub ciągi nieleksykalne (adresy internetowe itp.).

Definicja 7. Segmentacja tekstu – proces analizy tekstu polegający na podziale tekstu według przyjętych kryteriów. Kryteria podziału są zależne od wybranego języka, a także od wyniku, jaki chce się uzyskać. W przypadku podziału na wyrazy, jako kryterium podziału przyjmujemy znak spacji.

Przykład 3. Dopasowywanie wyrazów w korpusie polsko-angielskim za pomocą modelu 1.

Jako dane wejściowe przyjęto fragment korpusu polsko-angielskiego zawierającego 448 433 zdań. Pierwsze dwa zdania tego korpusu są postaci:

Action taken on Parliament's resolutions: see Minutes
Documents received: see Minutes

Działania podjęte w wyniku rezolucji Parlamentu: Patrz protokół,
Składanie dokumentów: patrz protokół

Po wykonaniu normalizacji (definicja 5.) i tokenizacji wyrazów (definicja 6.) otrzymujemy następujący wynik:

action taken on parliament resolutions see minutes
documents received see minutes

działania podjęte w wyniku rezolucji parlamentu patrz protokół
składanie dokumentów patrz protokół

Aby wyodrębnić wyrazy z poszczególnych zdań wykonuje się proces segmentacji tekstu (definicja 7.) z użyciem znaku spacji jako separatora. Następnie każdy wyraz ze zdania wejściowego jest łączony z każdym wyrazem ze zdania wyjściowego – w ten sposób powstaje bazowa macierz dopasowania dla pierwszej i drugiej pary zdań:

Aby zobrazować proces obliczeń na przykładzie maksymalnie uproszczonym, przyjmujemy że korpus ma postać:

see minutes
documents minutes

patrz protokół
dokumentów protokół

Możliwe dopasowania wyrazów dla pierwszej pary zdań w tym przypadku mają postać (należy przyjąć, że dopasowywany wyraz może nie mieć dopasowania – wtedy jest łączony z wyrazem pustym (ang. null)):

(0,0) : see-null, minutes-null;

(0,1) : see-null, minutes-*patrz*;

(0,2) : see-null, minutes-*protokół*;

(1,0) : see-*patrz*, minutes-null;

(1,1) : see-*patrz*, minutes-*patrz*;

(1,2) : see-*patrz*, minutes-*protokół*;

(2,0) : see-*protokół*, minutes-null;

(2,1) : see-*protokół*, minutes-*patrz*;

(2,2) : see-*protokół*, minutes-*protokół*;

Otrzymujemy 9 możliwych kombinacji powiązań między wyrazami.

W fazie inicjalizacji modelu 1., każda para wyrazów otrzymuje prawdopodobieństwo równe:

- dla pierwszej pary zdań $t(f \mid e) = \frac{1}{3}$ (gdyż korpus angielski zawiera 3 różne wyrazy),
- dla drugiej pary zdań $t(f \mid e) = \frac{1}{3}$.

W ten sposób suma prawdopodobieństw wszystkich dopasowań każdego wyrazu wynosi 1.

Parametry dla poszczególnych par zdań przyjmują następujące wartości:

- dla pierwszej pary zdań:
  - długość zdania wejściowego: $l = 2$
  - długość zdania wyjściowego: $m = 2$
  - liczba zdań w korpusie: $S = 2$

- dla drugiej pary zdań:
  - długość zdania wejściowego: $l = 2$
  - długość zdania wyjściowego: $m = 2$
  - liczba zdań w korpusie: $S = 2$

W kolejnym kroku algorytmu działania modelu 1. obliczamy prawdopodobieństwo tłumaczenia poszczególnych wyrazów. Obliczenia wykonamy dla pierwszej pary zdań (*see minutes – patrz protokół*).

$$\Pr_0(\mathbf{f}^{(1)},\mathbf{e}^{(1)}) = \sum_{a_1=0}^{2}\sum_{a_2=0}^{2}\prod_{j=1}^{2} t_0(f_j \mid e_{a_j}) =$$

$$= t_0(f_1 \mid e_0)t_0(f_2 \mid e_0) + t_0(f_1 \mid e_0)t_0(f_2 \mid e_1) + t_0(f_1 \mid e_0)t_0(f_2 \mid e_2) +$$
$$+ t_0(f_1 \mid e_1)t_0(f_2 \mid e_0) + t_0(f_1 \mid e_1)t_0(f_2 \mid e_1) + t_0(f_1 \mid e_1)t_0(f_2 \mid e_2) +$$
$$+ t_0(f_1 \mid e_2)t_0(f_2 \mid e_0) + t_0(f_1 \mid e_2)t_0(f_2 \mid e_1) + t_0(f_1 \mid e_2)t_0(f_2 \mid e_2) =$$
$$= 9 \cdot \frac{1}{3} \cdot \frac{1}{3} = 1$$

Obliczamy oczekiwaną częstość powiązań dla pary wyrazów *minutes – protokół:*

$$c(f \mid e; \mathbf{f}^{(1)}, \mathbf{e}^{(1)}) = \frac{\Pr_0(f, a \mid e)}{\Pr_0(\mathbf{f} \mid \mathbf{e})} \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_k}) =$$

$$= \frac{\sum_{a_1=0}^{2}\sum_{a_2=0}^{2}\prod_{j=1}^{2} t_0(f_j \mid e_{a_j})}{1} \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_k}) =$$

$$= t_0(f_1 \mid e_0)t_0(f_2 \mid e_0) \cdot 0 + t_0(f_1 \mid e_0)t_0(f_2 \mid e_1) \cdot 0 + t_0(f_1 \mid e_0)t_0(f_2 \mid e_2) \cdot 1 +$$
$$+ t_0(f_1 \mid e_1)t_0(f_2 \mid e_0) \cdot 0 + t_0(f_1 \mid e_1)t_0(f_2 \mid e_1) \cdot 0 + t_0(f_1 \mid e_1)t_0(f_2 \mid e_2) \cdot 1 +$$
$$+ t_0(f_1 \mid e_2)t_0(f_2 \mid e_0) \cdot 0 + t_0(f_1 \mid e_2)t_0(f_2 \mid e_1) \cdot 0 + t_0(f_1 \mid e_2)t_0(f_2 \mid e_2) \cdot 1 =$$
$$= 3 \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{3}$$

Dla drugiej pary zdań również otrzymujemy częstość powiązań równą w przybliżeniu 0,33, gdyż i w tym zdaniu para wyrazów *minutes – protokół* występuje jeden raz.

W ten sposób otrzymujemy całkowitą częstość powiązań pary wyrazów *minutes – protokół* dla całego rozpatrywanego korpusu równą:

$$c(\text{minutes} \mid \text{protokół}) = \sum_{i=1}^{2} c(\text{minutes} \mid \text{protokół}; \mathbf{f}^{(i)}, \mathbf{e}^{(i)}) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}.$$

Podobnie obliczamy całkowitą częstość powiązań dla wszystkich kombinacji wyrazów, otrzymując następującą tablicę wyników:

c(see|null)=0,33	c(minutes|null)=0,67	c(documents|null)=0,33
c(see|patrz) =0,33	c(minutes|patrz) =0,33	c(documents|patrz) =0
c(see|protokół) =0,33	c(minutes|protokół) =0,67	c(documents|protokół) =0,33
c(see|dokumentów) =0	c(minutes|dokumentów) =0,33	c(documents|dokumentów) =0,33

W ten sposób zakończyliśmy krok E algorytmu EM [13]. Prawdopodobieństwo tłumaczenia dla każdej pary wyrazów wyznaczamy z następującego wzoru:

$$t(f \mid e) = \frac{c(f \mid e)}{\sum_{f'} c(f' \mid e)},$$

czyli

$$t(\text{minutes} \mid \text{protokół}) = \frac{c(\text{minutes} \mid \text{protokół})}{\sum_{f} c(f \mid \text{protokół})} = \frac{\frac{2}{3}}{\frac{1}{3}+\frac{2}{3}+\frac{1}{3}} = \frac{\frac{2}{3}}{\frac{4}{3}} = \frac{1}{2}.$$

Wykonując obliczenia dla wszystkich par zdań, otrzymujemy:

t(see|null)=0,25	t(minutes|null)=0,5	t(documents|null)=0,25
t(see|patrz) =0,5	t(minutes|patrz) =0,5	t(documents|patrz) =0
t(see|protokół) =0,25	t(minutes|protokół) =0,5	t(documents|protokół) =0,25
t(see|dokumentów) =0	t(minutes|dokumentów) =0,5	t(documents|dokumentów) =0,5

Obliczając prawdopodobieństwa tłumaczenia dla wszystkich dopasowań wyrazów, mamy dane niezbędne do wyliczenia prawdopodobieństwa warunkowego dopasowania zdań korpusu dwujęzycznego dla pierwszej pary zdań:

$$\Pr_1(\mathbf{f}^{(1)} \mid \mathbf{e}^{(1)}) = \frac{\varepsilon(m \mid l)}{(l+1)^m} \sum_a \prod_{j=1}^{m} t\ (f_j \mid e_{a_j}) = \frac{1}{(2+1)^2} \sum_{a_1=0}^{2} \sum_{a_2=0}^{2} \prod_{j=1}^{2} t_0(f_j \mid e_{a_j}) =$$

$$= \frac{1}{9} \cdot t_0(f_1 \mid e_0) t_0(f_2 \mid e_0) + t_0(f_1 \mid e_0) t_0(f_2 \mid e_1) + t_0(f_1 \mid e_0) t_0(f_2 \mid e_2) +$$
$$+ t_0(f_1 \mid e_1) t_0(f_2 \mid e_0) + t_0(f_1 \mid e_1) t_0(f_2 \mid e_1) + t_0(f_1 \mid e_1) t_0(f_2 \mid e_2) +$$
$$+ t_0(f_1 \mid e_2) t_0(f_2 \mid e_0) + t_0(f_1 \mid e_2) t_0(f_2 \mid e_1) + t_0(f_1 \mid e_2) t_0(f_2 \mid e_2) =$$
$$= \frac{1}{9}(0{,}25 \cdot 0{,}5 + \cdot 0{,}25 \cdot 0{,}5 + 0{,}25 \cdot 0{,}5 +$$
$$+ 0{,}5 \cdot 0{,}5 + 0{,}5 \cdot 0{,}5 + 0{,}5 \cdot 0{,}5 +$$
$$+ 0{,}25 \cdot 0{,}5 + 0{,}25 \cdot 0{,}5 + 0{,}25 \cdot 0{,}5) =$$
$$= \frac{1}{9}(3 \cdot 0{,}125 + 3 \cdot 0{,}25 + 3 \cdot 0{,}125) = \frac{1}{9}(0{,}375 + 0{,}75 + 0{,}375) = 0{,}17$$

i dla drugiej pary zdań:

$$\Pr_1(\ \mathbf{f}^{(2)} \mid \mathbf{e}^{(2)}) = \frac{\varepsilon(m \mid l)}{(l+1)^m} \sum_a \prod_{j=1}^{m} t\ (f_j \mid e_{a_j}) = \frac{1}{(2+1)^2} \sum_{a_1=0}^{2} \sum_{a_2=0}^{2} \prod_{j=1}^{2} t_0(f_j \mid e_{a_j}) =$$

$$= \frac{1}{9} \cdot t_0(f_1 \mid e_0) t_0(f_2 \mid e_0) + t_0(f_1 \mid e_0) t_0(f_2 \mid e_1) + t_0(f_1 \mid e_0) t_0(f_2 \mid e_2) +$$
$$+ t_0(f_1 \mid e_1) t_0(f_2 \mid e_0) + t_0(f_1 \mid e_1) t_0(f_2 \mid e_1) + t_0(f_1 \mid e_1) t_0(f_2 \mid e_2) +$$
$$+ t_0(f_1 \mid e_2) t_0(f_2 \mid e_0) + t_0(f_1 \mid e_2) t_0(f_2 \mid e_1) + t_0(f_1 \mid e_2) t_0(f_2 \mid e_2) =$$
$$= \frac{1}{9}(0{,}25 \cdot 0{,}5 + \cdot 0{,}25 \cdot 0{,}5 + 0{,}25 \cdot 0{,}5 +$$
$$+ 0{,}5 \cdot 0{,}5 + 0{,}5 \cdot 0{,}5 + 0{,}5 \cdot 0{,}5 +$$
$$+ 0{,}25 \cdot 0{,}5 + 0{,}25 \cdot 0{,}5 + 0{,}25 \cdot 0{,}5) =$$
$$= \frac{1}{9}(3 \cdot 0{,}125 + 3 \cdot 0{,}25 + 3 \cdot 0{,}125) = \frac{1}{9}(0{,}375 + 0{,}75 + 0{,}375) = 0{,}17$$

Ustalanie nowych parametrów dla kolejnego kroku iteracji następuje w kroku M algorytmu. W tym kroku wykonuje się również normalizację otrzymanych wyników, umożliwiającą wykorzystanie ich w kolejnym kroku iteracji.

W wyniku działania pierwszej iteracji modelu 1. dla danego przypadku otrzymamy następujący plik wyjściowy macierzy dopasowania:

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.17
see minutes
NULL ({ 2 }) patrz ({ 1 }) protokół, ({ })
 # Sentence pair (2) source length 2 target length 2 alignment score : 0.17
documents minutes
NULL ({ 2 }) dokumentów ({ 1 }) protokół ({ })

Natomiast po pięciu krokach iteracji modelu 1. otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia:

t(see|null)=0,12	t(minutes|null)=0,76	t(documents|null)=0,12
t(see|patrz) =0,83	t(minutes|patrz) =0,16	t(documents|patrz) =0
t(see|protokół) =0,12	t(minutes|protokół) =0,76	t(documents|protokół) =0,12
t(see|dokumentów) =0	t(minutes|dokumentów) =0,16	t(documents|dokumentów) =0,83

Po pięciu iteracjach macierz wyjściowa ma postać:

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.542324
patrz protokół
NULL ({ 2 }) see ({ 1 }) minutes ({ })
 # Sentence pair (2) source length 2 target length 2 alignment score : 0.542324
dokumentów protokół
NULL ({ 2 }) documents ({ 1 }) minutes ({ })

Można zaobserwować, że model 1. po pięciu krokach poradził sobie dobrze

z dopasowaniem wyrazów *see-patrz i documents-documentów*, gorzej wygląda sytuacja z dopasowaniem *minutes-protokół*, gdyż z takim samym prawdopodobieństwem słowo *minutes* zostało powiązane z wyrazem pustym.

Model 2

W modelu 2. w czasie obliczania prawdopodobieństwa powiązania dwóch wyrazów bierze się pod uwagę pozycję wyrazów łączonych i długość zdań. Dodatkowo prawdopodobieństwo to zależy od kolejności występowania łączonych wyrazów w zdaniu.

W tym modelu zastosowano drugi rodzaj prawdopodobieństwa – prawdopodobieństwo dopasowania pozycji a. Jest to prawdopodobieństwo, że dla podanej długości obu zdań, wyraz docelowy na pozycji j jest dopasowany z wyrazem źródłowym na pozycji a_j. Prawdopodobieństwo warunkowe powiązania zdań $\mathbf{e}$ i $\mathbf{f}$ przyjmuje następującą postać:

$$\Pr(\mathbf{f} \mid \mathbf{e}) = \varepsilon(m \mid l) \sum_{a} \prod_{j=1}^{m} (t(f_j \mid e_{a_j}) a(a_j \mid j, m, l)). \tag{10}$$

Równania (4) i (5) dla modelu 2. nie ulegają zmianie, natomiast nowy parametr - prawdopodobieństwo dopasowania pozycji ma postać:

$$a(i \mid j, m, l) = \frac{\sum_{S=1}^{S} c(i \mid j, m, l; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}{\sum_{i'} \sum_{S=1}^{S} c(i' \mid j, m, l; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}, \tag{11}$$

gdzie

$$c(i \mid j, m, l; \mathbf{f}, \mathbf{e}) = \sum_{a} \Pr(a \mid \mathbf{f}, \mathbf{e}) \delta(i, a_j), \tag{12}$$

$$\Pr(a \mid \mathbf{f}, \mathbf{e}) = \frac{\prod_{j=1}^{m} (t(f_j \mid e_{a_j}) a(a_j, j, m, l))}{\sum_{a} \prod_{j=1}^{m} (t(f_j \mid e_{a_j}) a(a_j, j, m, l))}, \tag{13}$$

gdzie t jest prawdopodobieństwem tłumaczenia obliczonym podobnie jak w modelu 1.; a jest prawdopodobieństwem dopasowania pozycji.

Suma po długości zdania wejściowego prawdopodobieństw a sumuje się do 1:

$$\sum_{i=0}^{l} a(i \mid j, m, l) = 1. \tag{14}$$

Jako przykład opisujący działanie modelu 2. ponownie zostanie wykorzystany fragment korpusu polsko-angielskiego.

Przykład 4. Dopasowywanie wyrazów w korpusie polsko-angielskim za pomocą modelu 2.

Jako dane wejściowe przyjęto korpus polsko-angielski utworzony w przykładzie 3.:

see minutes
documents minutes

patrz protokół
dokumentów protokół

W fazie inicjalizacji modelu 2. każda para wyrazów otrzymuje prawdopodobieństwo równe (podobnie jak w modelu 1.):

- dla pierwszej pary zdań $t(f \mid e) = \frac{1}{3}$ (gdyż korpus angielski zawiera 3 różne wyrazy),
- dla drugiej pary zdań $t(f \mid e) = \frac{1}{3}$.

Parametry dla poszczególnych par zdań przyjmują następujące wartości:

- dla pierwszej pary zdań:
  - długość zdania wejściowego: $l = 2$
  - długość zdania wyjściowego: $m = 2$
  - liczba zdań w korpusie: $S = 2$
- dla drugiej pary zdań:
  - długość zdania wejściowego: $l = 2$
  - długość zdania wyjściowego: $m = 2$
  - liczba zdań w korpusie: $S = 2$

Prawdopodobieństwo dopasowania pozycji (interpretowane jako prawdopodobieństwo, że dla zdania f o długości m i dla zdania e o długości l, wyraz na i-tej pozycji jest dopasowany z wyrazem f na j-tej pozycji) w fazie inicjalizacji dla każdego dopasowania jest jednakowe i wynosi:

$a(i \mid j, m, l) = \frac{2}{3}$, (gdyż mamy dwa zdania o jednakowej długości i każdy wyraz ma trzy możliwe dopasowania – wliczając dopasowanie z wyrazem pustym)

Obliczamy częstość powiązań dla pary wyrazów *minutes – protokół:*

$$c(f \mid e;\mathbf{f}^{(1)},\mathbf{e}^{(1)}) = \sum_{a} \Pr(a \mid \mathbf{f},\mathbf{e}) \sum_{i,j} \delta(f, f_j)\delta(e, e_i) =$$

$$= \sum_{j=1}^{m}\sum_{i=0}^{l} \frac{t(f \mid e)a(i \mid j,m,l)\delta(f, f_i)\delta(e, e_i)}{t(f_j \mid e_0)a(0 \mid j,m,l) + \ldots + t(f_j \mid e_l)a(l \mid j,m,l)} =$$

$$= \frac{\frac{1}{3}\cdot\frac{2}{3}(0+0+1+0+0+1+0+0+1)}{\frac{1}{3}\cdot\frac{2}{3}\cdot 6} = \frac{\frac{2}{3}}{2} = \frac{1}{3}$$

Dla drugiej pary zdań również otrzymujemy częstość powiązań równą w przybliżeniu 0,33, gdyż i w tym zdaniu para wyrazów *minutes – protokół* występuje jeden raz.

W ten sposób otrzymujemy całkowitą częstość powiązań pary wyrazów *minutes - protokół* równą:

$$c(\text{mintuses} \mid \text{protokół}) = \sum_{i=1}^{2} c(\text{minutes} \mid \text{protokół};\mathbf{f}^{(i)},\mathbf{e}^{(i)}) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}.$$

Prawdopodobieństwo tłumaczenia obliczamy ze wzoru:

$$t(f \mid e) = \frac{\sum_{S=1}^{S} c(f \mid e;\mathbf{f}^{(s)},\mathbf{e}^{(s)})}{\sum_{f'}\sum_{S=1}^{S} c(f' \mid e;\mathbf{f}^{(s)},\mathbf{e}^{(s)})},$$

$$\text{czyli } t(\text{minutes} \mid \text{protokół}) = \frac{c(\text{minutes} \mid \text{protokół})}{\sum_{f} c(f \mid \text{protokół})} = \frac{\frac{2}{3}}{\frac{1}{3}+\frac{2}{3}+\frac{1}{3}} = \frac{\frac{2}{3}}{\frac{4}{3}} = \frac{1}{2}.$$

W pierwszej iteracji modelu 2. otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia i prawdopodobieństwa dopasowania pozycji :

t(see|null)=0,25	t(minutes|null)=0,5	t(documents|null)=0,25
t(see|patrz) =0,5	t(minutes|patrz) =0,5	t(documents|patrz) =0,00
t(see|protokół) =0,25	t(minutes|protokół) =0,5	t(documents|protokół) =0,25
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,5	t(documents|dokumentów) =0,5

a(0|1,2,2)=0,66	a(0|2,2,2)= 0,66
a(1|1,2,2)= 0,66	a(1|2,2,2)= 0,66
a(2|1,2,2)= 0,66	a(2|2,2,2)= 0,66

W wyniku działania pierwszej iteracji modelu 2. dla danego przypadku otrzymamy następujący plik wyjściowy macierzy dopasowania:

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.25
patrz protokół
NULL ({ 2 }) see ({ 1 }) minutes ({ })
 # Sentence pair (2) source length 2 target length 2 alignment score : 0.25
dokumentów protokół
NULL ({ 2 }) documents ({ 1 }) minutes ({ })

W piątej iteracji modelu 2. otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia i prawdopodobieństwa dopasowania pozycji:

t(see|null)=0,00	t(minutes|null)=0,99	t(documents|null)=0,00
t(see|patrz) =0,98	t(minutes|patrz) =0,01	t(documents|patrz) =0,00
t(see|protokół) =0,00	t(minutes|protokół) =0,99	t(documents|protokół) =0,00
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,01	t(documents|dokumentów) =0,98

a(0|1,2,2)=0,00	a(0|2,2,2)=0,99
a(1|1,2,2)=1,99	a(1|2,2,2)=0,03
a(2|1,2,2)=0,00	a(2|2,2,2)=0,99

Natomiast wynikowa macierz dopasowania ma postać:

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.542324
patrz protokół
NULL ({ 2 }) see ({ 1 }) minutes ({ })
 # Sentence pair (2) source length 2 target length 2 alignment score : 0.542324
dokumentów protokół
NULL ({ 2 }) documents ({ 1 }) minutes ({ })

Możemy zaobserwować, że nadal wyraz *minutes* ma duże prawdopodobieństwo tłumaczenia jako wyraz pusty i równe z prawdopodobieństwem tłumaczenia jako *protokół*. To powoduje, że domyślnie wyraz ten zostanie powiązany z wyrazem pustym. W porównaniu z modelem 1., w modelu obserwujemy zwiększenie prawdopodobieństwa tłumaczenia dla poprawnych powiązań wyrazów, dzięki czemu mniejsza liczba iteracji pozwala uzyskiwać poprawne dopasowania.

Model HMM

Model HMM jest wykorzystywany zamiast modelu 2. dopasowywania wyrazów. Wyniki wygenerowane przez ten model są znacząco lepsze od wyników modelu 2. Wykorzystuje on następujące równanie na prawdopodobieństwo warunkowe powiązania dwóch zdań:

$$\Pr(\mathbf{f} \mid \mathbf{e}) = \varepsilon(m \mid l) \sum_{a} \prod_{j=1}^{m} (t(f_j \mid e_{a_j}) a(a_{j-1}, l)), \tag{15}$$

które jest bardzo zbliżone do równania (10). Prawdopodobieństwo dopasowania pozycji a_j na pozycji j zależy od poprzedniego dopasowania pozycji a_{j-1}. Dzięki takiemu zabiegowi model dopasowania został przekształcony w model Markowa pierwszego rzędu. Prawdopodobieństwo dopasowania pozycji w takim przypadku jest zdefiniowane następująco:

$$a(i \mid i', l) = \frac{\sum_{S=1}^{S} c(i \mid i', l; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}{\sum_{i''} \sum_{S=1}^{S} c(i'' \mid i', l; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}, \tag{16}$$

$$c(i \mid i', l; \mathbf{f}, \mathbf{e}) = \sum_{a} \Pr(a \mid \mathbf{f}, \mathbf{e}) \sum_{j} (\delta(i', a_{j-1}) \delta(i, a_j)). \tag{17}$$

Prawdopodobieństwo tłumaczenia t i powiązane miary pozostają takie same jak w modelu 1. i 2. – równanie (5) i (6). Natomiast równanie $\Pr(a \mid \mathbf{f}, \mathbf{e})$ przyjmuje następującą postać:

$$\Pr(a \mid \mathbf{f}, \mathbf{e}) = \frac{\prod_{j=1}^{m} (t(f_j \mid e_{a_j}) a(a_j \mid a_{j-1}, l))}{\sum_{a} \prod_{j=1}^{m} ((t(f_j \mid e_{a_j}) a(a_j \mid a_{j-1}, l))}. \tag{18}$$

Jako przykład opisujący działanie modelu Markowa ponownie zostanie wykorzystany fragment korpusu polsko-angielskiego.

Przykład 5. Dopasowywanie wyrazów w korpusie polsko-angielskim za pomocą modelu HMM

Jako dane wejściowe przyjęto korpus polsko-angielski utworzony w przykładzie 3.:

see minutes
documents minutes

patrz protokół
dokumentów protokół

W fazie inicjalizacji modelu HMM para wyrazów otrzymuje prawdopodobieństwo równe (podobnie jak w modelu 1.):

- dla pierwszej pary zdań $t(f \mid e) = \frac{1}{3}$ (gdyż korpus angielski zawiera 3 różne wyrazy),
- dla drugiej pary zdań $t(f \mid e) = \frac{1}{3}$.

Parametry dla poszczególnych par zdań przyjmują następujące wartości:

- dla pierwszej pary zdań:
  - długość zdania wejściowego: $l = 2$
  - długość zdania wyjściowego: $m = 2$
  - liczba zdań w korpusie: $S = 2$
- dla drugiej pary zdań:
  - długość zdania wejściowego: $l = 2$
  - długość zdania wyjściowego: $m = 2$
  - liczba zdań w korpusie: $S = 2$

Prawdopodobieństwo dopasowania pozycji (w tym modelu prawdopodobieństwo to zależy od prawdopodobieństwa wyznaczonego w poprzedniej iteracji) w fazie inicjalizacji dla każdego dopasowania jest jednakowe i wynosi:

$a(i \mid i', l) = \frac{2}{3}$ (gdyż mamy dwa zdania o jednakowej długości i każdy wyraz ma trzy możliwe dopasowania – wliczając dopasowanie z wyrazem pustym).

Obliczamy częstość powiązań dla pary wyrazów *minutes – protokół:*

$$c(i \mid i', l; \mathbf{f}^{(1)}, \mathbf{e}^{(1)}) = \Pr(a \mid \mathbf{f}, \mathbf{e}) \sum_j \delta(i', a_{j-1}) \delta(i, a_j) =$$

$$= \frac{\prod_{j=1}^{m} t(f_j \mid e_{a_j}) a(a_j \mid a_{j-1}, l) \sum_j \delta(i', a_{j-1}) \delta(i, a_j)}{\sum_a \prod_{j=1}^{m} t(f_j \mid e_{a_j}) a(a_j \mid a_{j-1}, l)} =$$

$$= \frac{\prod_{j=1}^{2} \frac{1}{3} \cdot \frac{2}{3} \cdot \sum_j \delta(i', a_{j-1}) \delta(i, a_j)}{\sum_a \prod_{j=1}^{2} \frac{1}{3} \cdot \frac{2}{3}} =$$

$$= \frac{\prod_{j=1}^{2} \frac{2}{9} \cdot \sum_j \delta(i', a_{j-1}) \delta(i, a_j)}{9 \cdot \frac{2}{9} \cdot \frac{2}{9}} =$$

$$= \frac{(\frac{2}{9} \cdot \sum_j \delta(i', a_0) \delta(i, a_1))^2}{\frac{4}{9}} =$$

$$= \frac{9}{4} \cdot (\frac{2}{9} \cdot (\delta(i', a_0) \delta(i, a_1) + \delta(i', a_1) \delta(i, a_2))^2 =$$

$$= \frac{9}{4} \cdot (\frac{2}{9} \cdot (0+1))^2 = \frac{9}{4} \cdot \frac{4}{81} = \frac{1}{9}$$

Dla drugiej pary zdań również otrzymujemy częstość powiązań równą $\frac{1}{9}$, gdyż i w tym zdaniu para wyrazów *minutes – protokół* występuje jeden raz na tym samym miejscu w zdaniu.

W ten sposób otrzymujemy całkowitą częstość powiązań pary wyrazów *minutes – protokół* równą:

$$c(\text{minutes} \mid \text{protokół}) = \sum_{i=1}^{2} c(\text{minutes} \mid \text{protokół}; \mathbf{f}^{(i)}, \mathbf{e}^{(i)}) = \frac{1}{9} + \frac{1}{9} = \frac{2}{9}.$$

Prawdopodobieństwo tłumaczenia obliczamy ze wzoru:

$$t(f \mid e) = \frac{\sum_{S=1}^{S} c(f \mid e; \mathbf{f}^{(s)}, ^{(s)})}{\sum_{f'} \sum_{S=1}^{S} c(f' \mid e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})},$$

czyli $t(\text{minutes}\,|\,\text{protokół}) = \dfrac{c(\text{minutes}\,|\,\text{protokół})}{\sum_f c(f\,|\,\text{protokół})} = \dfrac{\frac{2}{9}}{\frac{1}{9}+\frac{2}{9}+\frac{1}{9}} = \dfrac{\frac{2}{9}}{\frac{4}{9}} = \dfrac{1}{2}$.

W pierwszej iteracji modelu HMM otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia i prawdopodobieństwa dopasowania pozycji:

t(see|null)=0,25	t(minutes|null)=0,5	t(documents|null)=0,25
t(see|patrz) =0,5	t(minutes|patrz) =0,5	t(documents|patrz) =0,00
t(see|protokół) =0,25	t(minutes|protokół) =0,5	t(documents|protokół) =0,25
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,5	t(documents|dokumentów) =0,5

a(0|1,2,2)=0,33	a(0|2,2,2)= 0,33
a(1|1,2,2)= 0,33	a(1|2,2,2)= 0,33
a(2|1,2,2)= 0,33	a(2|2,2,2)= 0,33

W wyniku działania pierwszej iteracji modelu HMM dla danego przypadku otrzymamy następujący plik wyjściowy macierzy dopasowania:

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.00694444
patrz protokół
NULL ({ 2 }) see ({ 1 }) minutes ({ })
 # Sentence pair (2) source length 2 target length 2 alignment score : 0.00694444
dokumentów protokół
NULL ({ 2 }) documents ({ 1 }) minutes ({ })

W piątej iteracji modelu HMM otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia i prawdopodobieństwa dopasowania pozycji:

t(see|null)=0,08	t(minutes|null)=0,84	t(documents|null)=0,08
t(see|patrz) =0,98	t(minutes|patrz) =0,01	t(documents|patrz) =0,00
t(see|protokół) =0,00	t(minutes|protokół) =0,99	t(documents|protokół) =0,00
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,01	t(documents|dokumentów) =0,98

a(0|1,2,2)=0,00	a(0|2,2,2)=0,03
a(1|1,2,2)=0,99	a(1|2,2,2)=0,01
a(2|1,2,2)=0,00	a(2|2,2,2)=0,96

Natomiast wynikowa macierz dopasowania wygląda następująco:

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.443452
patrz protokół
NULL ({ }) see ({ 1 }) minutes ({ 2 })
 # Sentence pair (2) source length 2 target length 2 alignment score : 0.443452

dokumentów protokół
NULL ({ }) documents ({ 1 }) minutes ({ 2 })

Jak można zauważyć, model HMM poprawnie dopasował wyrazy *minutes-protokół* (w przeciwieństwie do modelu 2.).

W modelu 3., 4. i 5. w fazie inicjalizacji oblicza się ilość wyrazów ze zdania docelowego, które będą połączone z wyrazem e ze zdania źródłowego. Zmienną tą nazywamy mnożnikiem (ang. fertility) wyrazu zdania **e** i oznaczamy Φ_e. Następnie tworzy się listę wyrazów docelowych, które są połączone z danym wyrazem zdania źródłowego **e**. Tą listę nazywamy tabletem *(ang. tablet)* τ wyrazu zdania źródłowego e.

Definicja 8. Model niepełny (ang. model deficiency) – model dopasowania wyrazów, w którym część prawdopodobieństwa przypisana jest zdarzeniom nieistotnym (niepożądanym).

Model 3

W modelu 3. prawdopodobieństwo powiązania wyrazów zależy od pozycji łączonych wyrazów i długości dopasowywanych zdań. Model ten jest niepełny (definicja 8.).

Prawdopodobieństwo warunkowe powiązania dwóch wyrazów w zdaniach **e** i **f** dla każdej pary wyrazów w modelu 3. ma następującą postać:

$$\Pr(\mathbf{f} \mid \mathbf{e}) = \sum_{a} \binom{m-\Phi_0}{\Phi_0} p_0^{m-2\Phi_0} p_1^{\Phi_0} \prod_{i=1}^{l} \Phi_i! n(\Phi_i \mid e_i) \times \prod_{j=1}^{m} t(f_j \mid e_{a_j}) d(j \mid a_j, m, l), \tag{19}$$

gdzie Φ_i oznacza mnożnik wyrazu zdania **e**, a p_1, p_2 są liczbami nieujemnymi, które sumują się do jedynki. Natomiast $d(j \mid i, m, l)$ jest prawdopodobieństwem zniekształcenia (ang. distortion probabilities) opisanym wzorem:

$$d(j \mid i, m, l) = \mu_{iml}^{-1} \sum_{s=1}^{S} c(j \mid i, m, l; \mathbf{f}^{(s)}, \mathbf{e}^{(s)}), \tag{20}$$

gdzie μ_{iml}^{-1} jest czynnikiem normalizującym.

$$c(j \mid i, m, l; \mathbf{f}, \mathbf{e}) = \sum_{a} \Pr(a \mid \mathbf{e}, \mathbf{f}) \delta(i, a_j), \tag{21}$$

$$\Pr(a \mid \mathbf{e}, \mathbf{f}) = \binom{m-\phi_0}{\phi_0} p_0^{m-2\phi_0} p_1^{\phi_0} \prod_{i=1}^{l} \phi_i! n(\phi_i \mid e_i) x \prod_{j=1}^{m} t(f_j \mid e_{a_j}) d(j \mid a_j, m, l) . \quad (22)$$

Prawdopodobieństwo tłumaczenia wyrazów jest opisane wzorem:

$$t(f \mid e) = \frac{\sum_{S=1}^{S} c(f \mid e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}{\sum_{f'} \sum_{S=1}^{S} c(f' \mid e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}, \quad (23)$$

$$c(f \mid e; \mathbf{f}, \mathbf{e}) = \sum_{a} \Pr(a \mid \mathbf{f}, \mathbf{e}) \sum_{i,j} \delta(f, f_j) \delta(e, e_i) . \quad (24)$$

Prawdopodobieństwo mnożności opisane jest wzorem:

$$n(\phi \mid e) = \nu_e^{-1} \sum_{s=1}^{S} c(\phi \mid e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)}), \quad (25)$$

gdzie ν_e jest stałą normalizacyjną.

Równania (20), (22) i (25) spełniają następujące zależności:

$$\sum_{f} t(f \mid e) = 1, \sum_{j} d(j \mid i, m, l) = 1, \sum_{\phi} n(\phi \mid e) = 1, \quad (26)$$

$$c(\phi \mid e; \mathbf{f}, \mathbf{e}) = \sum_{a} \Pr(a \mid \mathbf{e}, \mathbf{f}) \sum_{i=1}^{l} \delta(\phi, \phi_i) \delta(e, e_i) . \quad (27)$$

Jako przykład opisujący działanie modelu 3. ponownie zostanie wykorzystany fragment korpusu polsko-angielskiego.

Przykład 6. Dopasowywanie wyrazów w korpusie polsko-angielskim za pomocą modelu 3.

Jako dane wejściowe przyjęto korpus polsko-angielski utworzony w przykładzie 3.:

see minutes
documents minutes

patrz protokół
dokumentów protokół

Dopasowywanie wyrazów za pomocą modelu 3. bazuje na wynikach otrzymanych z poprzednich iteracji modelu 2. lub HMM.

Po wykonaniu jednej iteracji modelu 2. otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia i prawdopodobieństwa dopasowania pozycji:

t(see|null)=0,25	t(minutes|null)=0,5	t(documents|null)=0,25
t(see|patrz) =0,5	t(minutes|patrz) =0,5	t(documents|patrz) =0,00
t(see|protokół) =0,25	t(minutes|protokół) =0,5	t(documents|protokół) =0,25
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,5	t(documents|dokumentów) =0,5

a(0|1,2,2)=0,66	a(0|2,2,2)= 0,66
a(1|1,2,2)= 0,66	a(1|2,2,2)= 0,66
a(2|1,2,2)= 0,66	a(2|2,2,2)= 0,66

Parametry dla poszczególnych par zdań przyjmują następujące wartości:

- dla pierwszej pary zdań:
  - długość zdania wejściowego: $l = 2$
  - długość zdania wyjściowego: $m = 2$
  - liczba zdań w korpusie: $S = 2$
- dla drugiej pary zdań:
  - długość zdania wejściowego: $l = 2$
  - długość zdania wyjściowego: $m = 2$
  - liczba zdań w korpusie: $S = 2$

Aby obliczyć prawdopodobieństwo tłumaczenia ze wzoru

$$t(f \mid e) = \frac{\sum_{S=1}^{S} c(f \mid e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}{\sum_{f'} \sum_{S=1}^{S} c(f' \mid e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}$$

, musimy obliczyć częstość powiązań.

Dla pary wyrazów *minutes – protokół:*

$$c(f \mid e;\mathbf{f}^{(1)},\mathbf{e}^{(1)}) = \sum_a \Pr(a \mid \mathbf{f},\mathbf{e}) \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j}) =$$

$$= \sum_a \left(\binom{m-\phi_0}{\phi_0} p_0^{m-2\phi_0} p_1^{\phi_0} \left(\prod_{i=1}^{l} \phi_i! n(\phi_i \mid e_i) x \prod_{j=1}^{m} t(f_j \mid e_{a_j}) d(j \mid a_j, m, l) \right) \cdot \right.$$

$$\left. \cdot \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j}) \right) =$$

$$= \sum_a \left(\binom{2-1}{1} \cdot 0.67^{2-2\cdot 1} \cdot 0.33^1 \cdot (\phi_1! n(\phi_1 \mid e_1) \cdot \phi_2! n(\phi_2 \mid e_2) x \right.$$

$$\left. x((t(f_1 \mid e_{a_1}) d(1 \mid a_1, 2, 2) \cdot t(f_2 \mid e_{a_2}) d(2 \mid a_2, 2, 2)) \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j}) \right) =$$

$$= \sum_a (1 \cdot 1 \cdot 0.33 \cdot (2! \cdot n(2 \mid \text{see}) \cdot 2! \cdot n(2 \mid \text{minutes}) x$$

$$xt(\text{see} \mid e_{a_1}) \cdot d(1 \mid a_1, 2, 2) \cdot t(\text{minutes} \mid e_{a_2}) \cdot d(2 \mid a_2, 2, 2)) \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j})) =$$

$$= \sum_a (0.33 \cdot (2 \cdot 0.23 \cdot 2 \cdot 0.04) x (t(\text{see} \mid e_{a_1}) \cdot d(1 \mid a_1, 22) \cdot t(\text{minutes} \mid e_{a_2}) \cdot$$

$$\cdot d(2 \mid a_2, 2, 2)) \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j})) =$$

$$= \sum_a (0.33 \cdot ((0.46 \cdot t(\text{see} \mid e_{a_1}) \cdot d(1 \mid a_1, 2, 2) + 0.46 \cdot t(\text{minutes} \mid e_{a_2}) \cdot d(2 \mid a_2, 2, 2) +$$

$$+ 0.08 \cdot t(\text{see} \mid e_{a_1}) \cdot d(1 \mid a_1, 2, 2) + 0.08 \cdot t(\text{minutes} \mid e_{a_2}) \cdot d(2 \mid a_2, 2, 2)) \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j})) =$$

$$= \sum_a (0.33 \cdot ((0.54 \cdot t(\text{see} \mid e_{a_1}) \cdot d(1 \mid a_1, 2, 2) + 0.54 \cdot t(\text{minutes} \mid e_{a_2}) \cdot$$

$$\cdot d(2 \mid a_2, 2, 2)) \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j})) =$$

$$= 0.33 \cdot 0.54 \cdot (t(see \mid e_0) \cdot d(1 \mid 0, 2, 2) + t(\text{minutes} \mid e_0) \cdot d(2 \mid 0, 2, 2)) \cdot$$

$$\cdot (\delta(patrz, patrz)\delta(see, e_0) + \delta(patrz, protkol)\delta(see, e_0)) + \ldots$$

$$\ldots + t(see \mid e_2) \cdot d(1 \mid 2, 2, 2) + t(\text{minutes} \mid e_2) \cdot d(2 \mid 2, 2, 2)) \cdot$$

$$\cdot (\delta(patrz, patrz)\delta(see, e_2) + \delta(patrz, protkol)\delta(see, e_2)) =$$

$$= 0.1782 \cdot (t(see \mid null) \cdot d(1 \mid 0, 2, 2) + t(\text{minutes} \mid null) \cdot d(2 \mid 0, 2, 2)) \cdot$$

$$\cdot (\delta(patrz, patrz)\delta(see, null) + \delta(patrz, protkol)\delta(see, null)) + \ldots$$

$$\ldots + t(see \mid protokol) \cdot d(1 \mid 2, 22) + t(\text{minutes} \mid protokol) \cdot d(2 \mid 2, 2, 2)) \cdot$$

$$\cdot (\delta(patrz, patrz)\delta(see, \text{minutes}) + \delta(patrz, protkol)\delta(see, \text{minutes})) =$$

$$= 0.1782 \cdot (0 + 0 + (t(see \mid null) \cdot d(1 \mid 0, 2, 2) + t(\text{minutes} \mid protkol) \cdot d(2 \mid 2, 2, 2)) \cdot 1 + 0 + 0 +$$

$$+ (t(see \mid patrz) \cdot d(1 \mid 1, 2, 2) + t(\text{minutes} \mid protkol) \cdot d(2 \mid 2, 2, 2)) \cdot 1 + 0 + 0 +$$

$$+ (t(see \mid protokol) \cdot d(1 \mid 2, 2, 2) + t(\text{minutes} \mid protkol) \cdot d(2 \mid 2, 2, 2)) \cdot 1) =$$

$$0.1782 \cdot (0.25 \cdot 0.16 + 0.5 \cdot 0.85 + 0.5 \cdot 0.74 + 0.5 \cdot 0.85 + 0.25 \cdot 0.15 + 0.5 \cdot 0.85) =$$

$$\approx 0.3069$$

Dla drugiej pary zdań również otrzymujemy częstość powiązań równą w przybliżeniu 0,3069, gdyż i w tym zdaniu para wyrazów *minutes – protokół* występuje jeden raz, oba wyrazy znajdują się też w tym samym miejscu w zdaniu i długość łączonych wyrazów jest taka sama.

W ten sposób otrzymujemy całkowitą częstość powiązań pary wyrazów *minutes – protokół* równą:

$$c(\text{minutes}\,|\,\text{protokół}) = \sum_{i=1}^{2} c(\text{minutes}\,|\,\text{protokoł};\mathbf{f}^{(i)},\mathbf{e}^{(i)}) \approx 0{,}3069 + 0{,}3069 \approx 0{,}61$$

Prawdopodobieństwo tłumaczenia obliczamy ze wzoru:

$$t(f\,|\,e) = \frac{\sum_{S=1}^{S} c(f\,|\,e;\mathbf{f}^{(s)},\mathbf{e}^{(s)})}{\sum_{f'}\sum_{S=1}^{S} c(f'|\,e;\mathbf{f}^{(s)},\mathbf{e}^{(s)})},$$

$$\text{czyli } t(\text{minutes}\,|\,\text{protokol}) = \frac{c(\text{minutes}\,|\,\text{protokol})}{\sum_{f} c(f\,|\,\text{protokol})} = \frac{0{,}61}{0{,}055 + 0{,}61 + 0{,}055} = \frac{0{,}61}{0{,}72} \approx 0{,}85\,.$$

W pierwszej iteracji modelu 3. otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia, prawdopodobieństwa dopasowania pozycji, prawdopodobieństwa zniekształcenia i prawdopodobieństwa mnożności:

t(see|null)=0,08	t(minutes|null)=0,84	t(documents|null)=0,08
t(see|patrz) =0,74	t(minutes|patrz) =0,26	t(documents|patrz) =0,00
t(see|protokół) =0,08	t(minutes|protokół) =0,85	t(documents|protokół) =0,08
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,26	t(documents|dokumentów) =0,74

a(0|1,2,2)=0,01	a(0|2,2,2)= 0,07
a(1|1,2,2)= 0,88	a(1|2,2,2)= 0,30
a(2|1,2,2)= 0,11	a(2|2,2,2)= 0,63

d(1|0,2,2)=0,16	d(2|0,2,2)= 0,84
d(1|1,2,2)= 0,74	d(2|1,2,2)= 0,26
d(1|2,2,2)= 0,15	d(2|2,2,2)= 0,85

n(0|see)=0,05	n(0|minutes)=0,29	n(0|documents)=0,05
n(1|see)=0,72	n(1|minutes)=0,67	n(1|documents)=0,72
n(2|see)=0,23	n(2|minutes)=0,04	n(2|documents)=0,23
n(3|see)=0,00	n(3|minutes)=0,00	n(3|documents)=0,00
n(4|see)=0,00	n(4|minutes)=0,00	n(4|documents)=0,00
n(5|see)=0,00	n(5|minutes)=0,00	n(5|documents)=0,00

n(6|see)=0,00	n(6|minutes)=0,00	n(6|documents)=0,00
n(7|see)=0,00	n(7|minutes)=0,00	n(7|documents)=0,00
n(8|see)=0,00	n(8|minutes)=0,00	n(8|documents)=0,00
n(9|see)=0,00	n(9|minutes)=0,00	n(9|documents)=0,00

W wyniku działania pierwszej iteracji modelu 3. dla danego przypadku otrzymamy następujący plik wyjściowy macierzy dopasowania:

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.022102
patrz protokół
NULL ({ }) see ({ 1 }) minutes ({ 2 })
 # Sentence pair (2) source length 2 target length 2 alignment score : 0.022102
dokumentów protokół
NULL ({ }) documents ({ 1 }) minutes ({ 2 })

W piątej iteracji modelu 3. otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia, prawdopodobieństwa dopasowania pozycji, prawdopodobieństwa zniekształcenia i prawdopodobieństwa mnożności:

t(see|null)=0,33	t(minutes|null)=0,33	t(documents|null)=0,33
t(see|patrz) =1,00	t(minutes|patrz) =0,00	t(documents|patrz) =0,00
t(see|protokół) =0,00	t(minutes|protokół) =1,00	t(documents|protokół) =0,00
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,00	t(documents|dokumentów) =1,00

a(0|1,2,2)=0,00	a(0|2,2,2)=0,99
a(1|1,2,2)=1,99	a(1|2,2,2)=0,03
a(2|1,2,2)=0,00	a(2|2,2,2)=0,99

d(1|0,2,2)=0,00	d(2|0,2,2)= 1,00
d(1|1,2,2)= 1,00	d(2|1,2,2)= 0,00
d(1|2,2,2)= 0,00	d(2|2,2,2)= 1,00

n(0|see)=0,00	n(0|minutes)=0,00	n(0|documents)=0,00
n(1|see)=0,99	n(1|minutes)=1,00	n(1|documents)=0,99
n(2|see)=0,00	n(2|minutes)=0,00	n(2|documents)=0,00
n(3|see)=0,00	n(3|minutes)=0,00	n(3|documents)=0,00
n(4|see)=0,00	n(4|minutes)=0,00	n(4|documents)=0,00
n(5|see)=0,00	n(5|minutes)=0,00	n(5|documents)=0,00
n(6|see)=0,00	n(6|minutes)=0,00	n(6|documents)=0,00
n(7|see)=0,00	n(7|minutes)=0,00	n(7|documents)=0,00
n(8|see)=0,00	n(8|minutes)=0,00	n(8|documents)=0,00
n(9|see)=0,00	n(9|minutes)=0,00	n(9|documents)=0,00

Natomiast wynikowa macierz dopasowania po pięciu iteracjach modelu 3. ma postać:

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.999999
patrz protokół
NULL ({ }) see ({ 1 }) minutes ({ 2 })
 # Sentence pair (2) source length 2 target length 2 alignment score : 0.999999
dokumentów protokół
NULL ({ }) documents ({ 1 }) minutes ({ 2 })

Możemy zaobserwować, że model 3. (w przeciwieństwie do modelu 2.) prawidłowo dopasował wyrazy *minutes – protokół*, nadając im prawdopodobieństwo tłumaczenia równe 1.

Model 4

W modelu 4. prawdopodobieństwo powiązania wyrazów zależy od prawdopodobieństwa dopasowania wyrazów obliczonego w poprzednim modelu i od pozycji innych wyrazów zdania docelowego, które są powiązane z tym samym wyrazem źródłowym. Zależy również, podobnie jak w modelu 3. od długości łączonych zdań. Główna różnica polega na sposobie wyznaczania prawdopodobieństwa zniekształcenia. W modelu 4. każde słowo jest zależne od klasy słowa wyznaczonej z otaczających wyrazów. Model ten jest również niepełny (definicja 8.).

Model 3. nie radzi sobie z tłumaczeniami, w których wyraz tłumaczenia (ze zdania docelowego) może występować w różnych miejscach w zdaniu. W modelu 4. ten problem rozwiązano.

Definicja 9. Tablet – pojęcie stosowane w algorytmach dopasowywania wyrazów, definiowane jako zbiór, w którym dla każdego wyrazu źródłowego ustala się mnożnik i listę wyrazów docelowych, z którymi łączy się dany wyraz źródłowy.

Definicja 10. Centrum – definiuje sufit średnich wartości pozycji słów z tabletu w zdaniu docelowym.

Definicja 11. Głowa – jest to wyraz z tabletu, dla którego pozycja w zdaniu docelowym jest najmniejsza.

W modelu 4. wykorzystujemy tablet (definicja 9.) zawierający centrum i głowę.

Prawdopodobieństwo warunkowe w modelu 4. ma następującą postać:

$$\Pr(\prod_{[i]1} = j \mid \pi_1^{[i]-1}, \tau_0^l, \phi_0^l, e) = d_1(j - \Theta_{i-1} \mid \mathrm{A}(e_{[i-1]}, \mathrm{B}(f_j)), \quad (28)$$

$$\Pr(\prod_{[i]k} = j \mid \pi_{[i]1}^{k-1}, \pi_1^{[i]-1}, \tau_0^l, \phi_0^l, e) = d_{>1}(j - \pi_{[i]k-1} \mid \mathrm{B}(f_j)), \quad (29)$$

gdzie Θ_i jest wspomnianą powyżej średnią wartością pozycji w zdaniu (centrum), A i B są funkcjami wyrazów zdania źródłowego i docelowego.

Pozostałe równania są zbliżone do równań dla modelu 3.

Jako przykład opisujący działanie modelu 4. wykorzystamy przykład zbliżony do tego przedstawionego w modelu 3. Wprowadzimy tutaj pewne modyfikacje ukazujące przewagę modelu 4. nad modelem 3.

Przykład 7. Dopasowywanie wyrazów w korpusie polsko-angielskim za pomocą modelu 4.

Jako dane wejściowe przyjęto korpus polsko-angielski:

see minutes
documents minutes

patrz protokół
protokół dokumentów

W porównaniu z korpusem z przykładu 6. zamieniono wyrażenie *dokumentów protokół* na *protokół dokumentów.* Po takiej modyfikacji wykonanie pięciu iteracji modelu 3. daje następujące prawdopodobieństwa tłumaczenia:

t(see|null)=0,33	t(minutes|null)=0,33	t(documents|null)=0,33
t(see|patrz) =0,5	t(minutes|patrz) =0,5	t(documents|patrz) =0,00
t(see|protokół) =0,33	t(minutes|protokół) =0,33	t(documents|protokół) =0,33
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,5	t(documents|dokumentów) =0,5

Jak widzimy, nie zostało tu rozstrzygnięte, z jakim wyrazem połączyć wyraz *minutes.*

Dopasowywanie wyrazów za pomocą modelu 4. bazuje na wynikach otrzymanych z poprzednich iteracji modelu 2., HMM oraz 3. Po wykonaniu jednej iteracji modelu 2. otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia i prawdopodobieństwa dopasowania pozycji:

t(see|null)=0,25	t(minutes|null)=0,5	t(documents|null)=0,25
t(see|patrz) =0,5	t(minutes|patrz) =0,5	t(documents|patrz) =0,00
t(see|protokół) =0,25	t(minutes|protokół) =0,5	t(documents|protokół) =0,25
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,5	t(documents|dokumentów) =0,5

a(0|1,2,2)=0,66	a(0|2,2,2)= 0,66
a(1|1,2,2)= 0,66	a(1|2,2,2)= 0,66
a(2|1,2,2)= 0,66	a(2|2,2,2)= 0,66

Parametry dla poszczególnych par zdań przyjmują następujące wartości:

- dla pierwszej pary zdań:
  - długość zdania wejściowego: $l = 2$
  - długość zdania wyjściowego: $m = 2$
  - liczba zdań w korpusie: $S = 2$
- dla drugiej pary zdań:
  - długość zdania wejściowego: $l = 2$
  - długość zdania wyjściowego: $m = 2$
  - liczba zdań w korpusie: $S = 2$

W pierwszej iteracji modelu 4. otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia, prawdopodobieństwa dopasowania pozycji, prawdopodobieństwa zniekształcenia i prawdopodobieństwa mnożności:

t(see|null)=0,08	t(minutes|null)=0,83	t(documents|null)=0,08
t(see|patrz) =0,68	t(minutes|patrz) =0,32	t(documents|patrz) =0,00
t(see|protokół) =0,12	t(minutes|protokół) =0,76	t(documents|protokół) =0,12
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,32	t(documents|dokumentów) =0,68

a(0|1,2,2)=0,04	a(0|2,2,2)= 0,04
a(1|1,2,2)= 0,61	a(1|2,2,2)= 0,61
a(2|1,2,2)= 0,36	a(2|2,2,2)= 0,36

d(1|0,2,2)=0,5	d(2|0,2,2)= 0,5
d(1|1,2,2)= 0,5	d(2|1,2,2)= 0,5
d(1|2,2,2)= 0,5	d(2|2,2,2)= 0,5

n(0|see)=0,06	n(0|minutes)=0,34	n(0|documents)=0,06
n(1|see)=0,67	n(1|minutes)=0,62	n(1|documents)=0,67
n(2|see)=0,27	n(2|minutes)=0,05	n(2|documents)=0,27
n(3|see)=0,00	n(3|minutes)=0,00	n(3|documents)=0,00
n(4|see)=0,00	n(4|minutes)=0,00	n(4|documents)=0,00

n(5|see)=0,00	n(5|minutes)=0,00	n(5|documents)=0,00
n(6|see)=0,00	n(6|minutes)=0,00	n(6|documents)=0,00
n(7|see)=0,00	n(7|minutes)=0,00	n(7|documents)=0,00
n(8|see)=0,00	n(8|minutes)=0,00	n(8|documents)=0,00
n(9|see)=0,00	n(9|minutes)=0,00	n(9|documents)=0,00

W wyniku działania pierwszej iteracji modelu 4. dla danego przypadku otrzymamy następujący plik wyjściowy macierzy dopasowania:

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.0161161
patrz protokół
NULL ({ }) see ({ 1 }) minutes ({ 2 })
 # Sentence pair (2) source length 2 target length 2 alignment score : 0.0161161
protokół dokumentów
NULL ({ }) documents ({ 2 }) minutes ({ 1 })

W piątej iteracji modelu 4. otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia, prawdopodobieństwa dopasowania pozycji, prawdopodobieństwa zniekształcenia i prawdopodobieństwa mnożności:

t(see|null)=0,33	t(minutes|null)=0,33	t(documents|null)=0,33
t(see|patrz) =1,00	t(minutes|patrz) =0,00	t(documents|patrz) =0,00
t(see|protokół) =0,00	t(minutes|protokół) =1,00	t(documents|protokół) =0,00
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,00	t(documents|dokumentów) =1,00

a(0|1,2,2)=0,00	a(0|2,2,2)=0,00
a(1|1,2,2)=0,5	a(1|2,2,2)=0,5
a(2|1,2,2)=0,5	a(2|2,2,2)=0,5

d(1|0,2,2)=0,5	d(2|0,2,2)= 0,5
d(1|1,2,2)= 1,00	d(2|1,2,2)= 0,5
d(1|2,2,2)= 0,5	d(2|2,2,2)= 0,5

n(0|see)=0,00	n(0|minutes)=0,00	n(0|documents)=0,00
n(1|see)=0,99	n(1|minutes)=1,00	n(1|documents)=0,99
n(2|see)=0,00	n(2|minutes)=0,00	n(2|documents)=0,00
n(3|see)=0,00	n(3|minutes)=0,00	n(3|documents)=0,00
n(4|see)=0,00	n(4|minutes)=0,00	n(4|documents)=0,00
n(5|see)=0,00	n(5|minutes)=0,00	n(5|documents)=0,00
n(6|see)=0,00	n(6|minutes)=0,00	n(6|documents)=0,00
n(7|see)=0,00	n(7|minutes)=0,00	n(7|documents)=0,00
n(8|see)=0,00	n(8|minutes)=0,00	n(8|documents)=0,00
n(9|see)=0,00	n(9|minutes)=0,00	n(9|documents)=0,00

Natomiast wynikowa macierz dopasowania po pięciu iteracjach modelu 4. ma postać:

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.404184
patrz protokół
NULL ({ }) see ({ 1 }) minutes ({ 2 })
 # Sentence pair (2) source length 2 target length 2 alignment score : 0.403795
protokół dokumentów
NULL ({ }) documents ({ 2 }) minutes ({ 1 })

W porównaniu z modelem 3. w tym przypadku wyraz *minutes* został poprawnie dopasowany z wyrazem *protokół*, co obrazuje przewagę modelu 4. w dopasowywaniu wyrazów, które znajdują się na różnych pozycjach w zdaniach.

Model 5

Model 5. jest zmodyfikowanym modelem 4. Dodano tu specjalne dopasowanie słów (ang. refined alignment) aby wyeliminować niepełność modelu (definicja 8.). W ten sposób liczba parametrów dopasowań znacznie zwiększa się w porównaniu do modelu 4.

I tak prawdopodobieństwo warunkowe dla modelu 5. przyjmuje następującą postać:

$$\Pr(\prod_{[i]1} = j \mid \pi_1^{[i]-1}, \tau_0^l, \phi_0^l, e) = d_1(\nu_j \mid \mathrm{B}(f_j), \nu_{\Theta_{i-1}}, \nu_m - \Theta_{[i]} + 1)(1 - \delta(\nu_j, \nu_{j-1})), \tag{30}$$

$$\begin{aligned}&\Pr(\prod_{[i]k} = j \mid \pi_{[i]1}^{k-1}, \pi_1^{[i]-1}, \tau_0^l, \phi_0^l, e) = \\ &= d_{>1}(\nu_j - \nu_{\pi_{[i]k-1}} \mid \mathrm{B}(f_j), \nu_m - \nu_{\pi_{[i]k-1}} - \phi_{[i]} + k)(1 - \delta(\nu_j, \nu_{j-1}))\end{aligned}, \tag{31}$$

gdzie ν_j oznacza liczbę wyrazów w zdaniu docelowym bez powiązań z wyrazami ze zdania źródłowego. Szczegóły dotyczące modelu 5 można znaleźć między innymi w pracy Brown'a [9].

Jako przykład opisujący działanie modelu 5. ponownie zostanie wykorzystany fragment korpusu polsko-angielskiego.

Przykład 8. Dopasowywanie wyrazów w korpusie polsko-angielskim za pomocą modelu 5

Jako dane wejściowe przyjęto korpus polsko-angielski utworzony w przykładzie 3.:

see minutes
documents minutes

patrz protokół
protokół dokumentów

Dopasowywanie wyrazów za pomocą modelu 5. wykorzystuje wyniki otrzymane z jednego z poprzednich modeli dopasowania. W przykładzie ponownie zostaną wykorzystane wyniki zwrócone przez model 2.

Po uruchomieniu modelu 2. otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia i prawdopodobieństwa dopasowania pozycji:

t(see|null)=0,08	t(minutes|null)=0,83	t(documents|null)=0,08
t(see|patrz) =0,68	t(minutes|patrz) =0,32	t(documents|patrz) =0,00
t(see|protokół) =0,12	t(minutes|protokół) =0,76	t(documents|protokół) =0,12
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,32	t(documents|dokumentów) =0,68

a(0|1,2,2)=0,66	a(0|2,2,2)= 0,66
a(1|1,2,2)= 0,66	a(1|2,2,2)= 0,66
a(2|1,2,2)= 0,66	a(2|2,2,2)= 0,66

Parametry dla poszczególnych par zdań przyjmują następujące wartości:

- dla pierwszej pary zdań:
  - długość zdania wejściowego: $l = 2$
  - długość zdania wyjściowego: $m = 2$
  - liczba zdań w korpusie: $S = 2$
- dla drugiej pary zdań:
  - długość zdania wejściowego: $l = 2$
  - długość zdania wyjściowego: $m = 2$
  - liczba zdań w korpusie: $S = 2$

Po pierwszej iteracji modelu 5. otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia, prawdopodobieństwa dopasowania pozycji, prawdopodobieństwa zniekształcenia i prawdopodobieństwa mnożności:

t(see|null)=0,08	t(minutes|null)=0,83	t(documents|null)=0,08
t(see|patrz) =0,68	t(minutes|patrz) =0,32	t(documents|patrz) =0,00
t(see|protokół) =0,12	t(minutes|protokół) =0,76	t(documents|protokół) =0,12
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,32	t(documents|dokumentów) =0,68

a(0|1,2,2)=0,04	a(0|2,2,2)= 0,04
a(1|1,2,2)= 0,61	a(1|2,2,2)= 0,61
a(2|1,2,2)= 0,36	a(2|2,2,2)= 0,36

d(1|0,2,2)=0,5	d(2|0,2,2)= 0,5
d(1|1,2,2)= 0,5	d(2|1,2,2)= 0,5
d(1|2,2,2)= 0,5	d(2|2,2,2)= 0,5

n(0|see)=0,06	n(0|minutes)=0,34	n(0|documents)=0,06
n(1|see)=0,67	n(1|minutes)=0,62	n(1|documents)=0,67
n(2|see)=0,27	n(2|minutes)=0,05	n(2|documents)=0,27
n(3|see)=0,00	n(3|minutes)=0,00	n(3|documents)=0,00
n(4|see)=0,00	n(4|minutes)=0,00	n(4|documents)=0,00
n(5|see)=0,00	n(5|minutes)=0,00	n(5|documents)=0,00
n(6|see)=0,00	n(6|minutes)=0,00	n(6|documents)=0,00
n(7|see)=0,00	n(7|minutes)=0,00	n(7|documents)=0,00
n(8|see)=0,00	n(8|minutes)=0,00	n(8|documents)=0,00
n(9|see)=0,00	n(9|minutes)=0,00	n(9|documents)=0,00

Plik wyjściowy macierzy dopasowania przyjmuje następującą postać:

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.0161161
patrz protokół
NULL ({ }) see ({ 1 }) minutes ({ 2 })
 # Sentence pair (2) source length 2 target length 2 alignment score : 0.0161161
protokół dokumentów
NULL ({ }) documents ({ 2 }) minutes ({ 1 })

Natomiast po pięciu iteracjach modelu 5. wartości prawdopodobieństwa tłumaczenia, prawdopodobieństwa dopasowania pozycji, prawdopodobieństwa zniekształcenia i prawdopodobieństwa mnożności mają następującą postać:

t(see|null)=0,33	t(minutes|null)=0,33	t(documents|null)=0,33
t(see|patrz) =1,00	t(minutes|patrz) =0,00	t(documents|patrz) =0,00
t(see|protokół) =0,00	t(minutes|protokół) =1,00	t(documents|protokół) =0,00
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,00	t(documents|dokumentów) =1,00

a(0|1,2,2)=0,00	a(0|2,2,2)= 0,00
a(1|1,2,2)= 0,5	a(1|2,2,2)= 0,5
a(2|1,2,2)= 0,5	a(2|2,2,2)= 0,5

d(1|0,2,2)=0,5	d(2|0,2,2)= 0,5
d(1|1,2,2)= 0,5	d(2|1,2,2)= 0,5
d(1|2,2,2)= 0,5	d(2|2,2,2)= 0,5

n(0|see)=0,00	n(0|minutes)=0,00	n(0|documents)=0,00
n(1|see)=0,99	n(1|minutes)=1,00	n(1|documents)=0,99
n(2|see)=0,00	n(2|minutes)=0,00	n(2|documents)=0,00
n(3|see)=0,00	n(3|minutes)=0,00	n(3|documents)=0,00
n(4|see)=0,00	n(4|minutes)=0,00	n(4|documents)=0,00
n(5|see)=0,00	n(5|minutes)=0,00	n(5|documents)=0,00
n(6|see)=0,00	n(6|minutes)=0,00	n(6|documents)=0,00
n(7|see)=0,00	n(7|minutes)=0,00	n(7|documents)=0,00
n(8|see)=0,00	n(8|minutes)=0,00	n(8|documents)=0,00
n(9|see)=0,00	n(9|minutes)=0,00	n(9|documents)=0,00

Wynikowa macierz dopasowania po pięciu iteracjach modelu 5. ma postać:

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.949995
patrz protokół
NULL ({ }) see ({ 1 }) minutes ({ 2 })
 # Sentence pair (2) source length 2 target length 2 alignment score : 0.949995
protokół dokumentów
NULL ({ }) documents ({ 2 }) minutes ({ 1 })

Model 5. podobnie jak model 4. poprawnie dopasowuje ze sobą wyrazy *minutes* - *protokół.* Jednak w tym przypadku punktacja dopasowania (ang. alignment score) jest bliska jeden. Wartość ta informuje o prawdopodobieństwie dobrego dopasowania wyrazów w danej sentencji. W tym przypadku algorytm dopasowania informuje, że jest praktycznie pewny tego dopasowania wyrazów. Ta właściwość ukazuje główną przewagę modelu 5. nad modelem 4. - algorytm szybciej uzyskuje zbieżność do 1. W związku z tym mniejsza ilość iteracji, a co za tym idzie mniejsza ilość czasu obliczeń, jest potrzebna, aby uzyskać trafne dopasowania wyrazów.

Model 6

Model 6. [12] jest kombinacją modelu HMM i modelu 4., wykorzystującą zalety obu modeli. Prawdopodobieństwo warunkowe dla tego modelu przyjmuje następującą postać:

$$\Pr(\mathbf{f}, a \mid \mathbf{e}) = \frac{\Pr_4(\mathbf{f}, a \mid \mathbf{e})^{\alpha} \cdot \Pr_{HMM}(\mathbf{f}, a \mid \mathbf{e})}{\sum_{a',\mathbf{f}'} \Pr_4(\mathbf{f}', a' \mid \mathbf{e})^{\alpha} \cdot \Pr_{HMM}(\mathbf{f}', a' \mid \mathbf{e})}, \tag{32}$$

gdzie α jest parametrem interpolacyjnym oznaczającym wagę modelu 4. względem modelu HMM [12]. Parametr α jest tak dobierany, aby zoptymalizować jakość dopasowania danych wyjściowych.

Jako przykład opisujący działanie modelu 6. kolejny raz wykorzystany zostanie fragment korpusu polsko-angielskiego.

Przykład 9. Dopasowywanie wyrazów w korpusie polsko-angielskim za pomocą modelu 6.

Jako dane wejściowe przyjęto korpus polsko-angielski utworzony w przykładzie 3.:

see minutes
documents minutes

patrz protokół
dokumentów protokół

Aby uruchomić dopasowywanie wyrazów za pomocą modelu 6. musimy wykonać przynajmniej jedną iterację modelu 4. i modelu HMM.

Dla modelu HMM po jednej iteracji otrzymujemy następujące wartości prawdopodobieństwa tłumaczenia i prawdopodobieństwa dopasowania pozycji:

t(see|null)=0,25	t(minutes|null)=0,5	t(documents|null)=0,25
t(see|patrz) =0,5	t(minutes|patrz) =0,5	t(documents|patrz) =0,00
t(see|protokół) =0,25	t(minutes|protokół) =0,5	t(documents|protokół) =0,25
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,5	t(documents|dokumentów) =0,5

a(0|1,2,2)=0,66	a(0|2,2,2)= 0,66
a(1|1,2,2)= 0,66	a(1|2,2,2)= 0,66
a(2|1,2,2)= 0,66	a(2|2,2,2)= 0,66

Natomiast, dodatkowo, po uruchomieniu iteracji modelu 4 otrzymujemy:

t(see|null)=0,19	t(minutes|null)=0,62	t(documents|null)=0,19
t(see|patrz) =0,57	t(minutes|patrz) =0,42	t(documents|patrz) =0,00
t(see|protokół) =0,17	t(minutes|protokół) =0,67	t(documents|protokół) =0,17
t(see|dokumentów) =0,00	t(minutes|dokumentów) =0,42	t(documents|dokumentów) =0,58

a(0|1,2,2)=0,79	a(0|2,2,2)= 0,13
a(1|1,2,2)= 0,76	a(1|2,2,2)= 0,56
a(2|1,2,2)= 0,16	a(2|2,2,2)= 0,32

Wynikowa macierz dla modelu 6 jest uzależniona od wyników dla modelu 4 i HMM. Jeśli wykonamy iterację modelu 6 dla jednej iteracji modelu 4 i jednej iteracji modelu HMM, to wyniki będą znacznie gorsze niż wyniki np. pięciu iteracji modelu 4. Dlatego, aby uzyskać

dobre wyniki, doświadczenia wykazały, że optymalną ilością iteracji jest pięć iteracji modelu 4. i pięć iteracji modelu 5. Dopiero wtedy można przejść do iteracji modelu 6.

1.5. Narzędzia dopasowania wyrazów

Opracowano wiele narzędzi służących do dopasowania wyrazów metodami statystycznymi, np.: BerkleyAligner [15], NATools [16], unl-aligner [17], K-vec++ [18]. Najbardziej popularnym, dostępnym darmowo (licencja GNU) narzędziem jest Giza++ [12]. W tym rozdziale przedstawię architekturę tego rozwiązania, a także jego rozwinięcia, jakie powstały w ostatnim dziesięcioleciu. Giza++ korzysta z szeregu modeli statystycznych, pozwalających dopasować wyrazy, będące swoim wzajemnym tłumaczeniem. Narzędzie można wywoływać z parametrami, których odpowiednie użycie podwyższa jakość dopasowania, bądź szybkość obliczeń.

Aby zwiększyć szybkość obliczeń, nie tracąc przy tym jakości, opracowano narzędzia MGiza++, PGiza++ i Chaski [19], [20], [21]]. Wykorzystują one wiele rdzeni współczesnych komputerów, pozwalając znacząco zredukować czas obliczeń.

1.5.1. Giza++, MGiza++, PGiza++, Chaski – architektura, opis działania

Giza++, MGiza++, PGiza++ i Chaski wymagają przygotowania szeregu plików wejściowych, niezbędnych w procesie dopasowania wyrazów. Są to pliki zawierające:

- odpowiedni format korpusów dwujęzycznych,
- klasy słów (ang. word classes),
- dodatkowe pliki – obliczenia wejściowe.

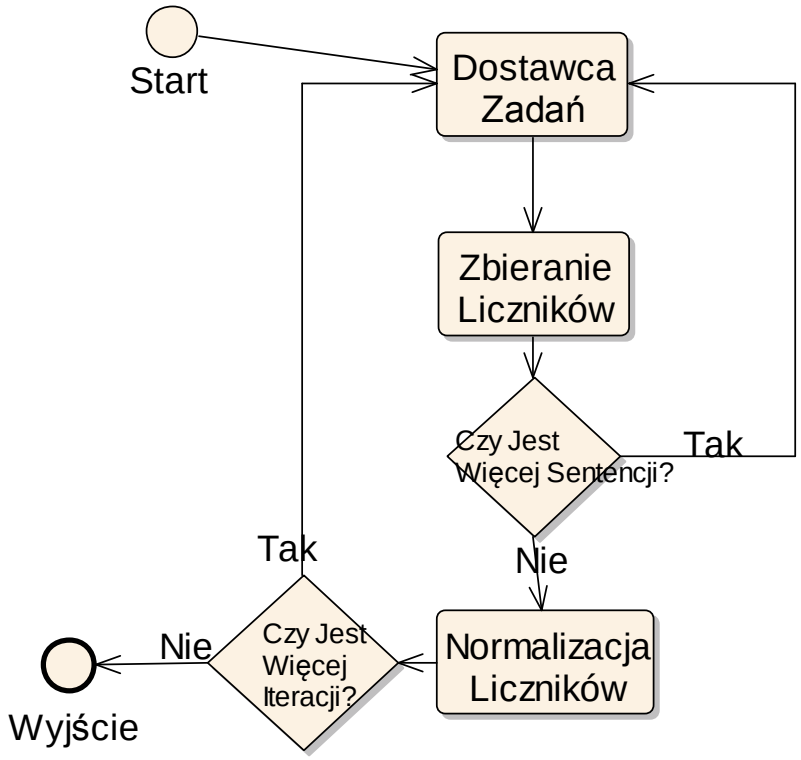
Po utworzeniu tych plików można uruchomić odpowiedni program dopasowywania wyrazów.

Giza++

Giza++ wykorzystuje pojedynczy procesor w fazie obliczeń. Stosowanie tego narzędzia związane jest z długim czasem oczekiwania – przy dużych korpusach danych czas liczony jest w godzinach, a nawet dniach.

Na rysunku 7. przedstawiono działanie algorytmu Giza++. Po uruchomieniu programu odpowiedni moduł (*Dostawca Zdań*) przydziela kolejne pary zdań: zdanie źródłowe i jego tłumaczenie do procesu dopasowywania zdań. Następuje proces dopasowywania wyrazów.

Krok ten jest powtarzany do momentu wyczerpania puli dostępnych zdań. Następuje unormowanie otrzymanych wyników. Uruchamiana jest kolejna iteracja danego modelu lub kolejny model statystyczny, w zależności od konfiguracji programu. Po wykonaniu wszystkich iteracji każdego modelu, program zwraca wyniki zawierające dopasowania wyrazów w danym korpusie.



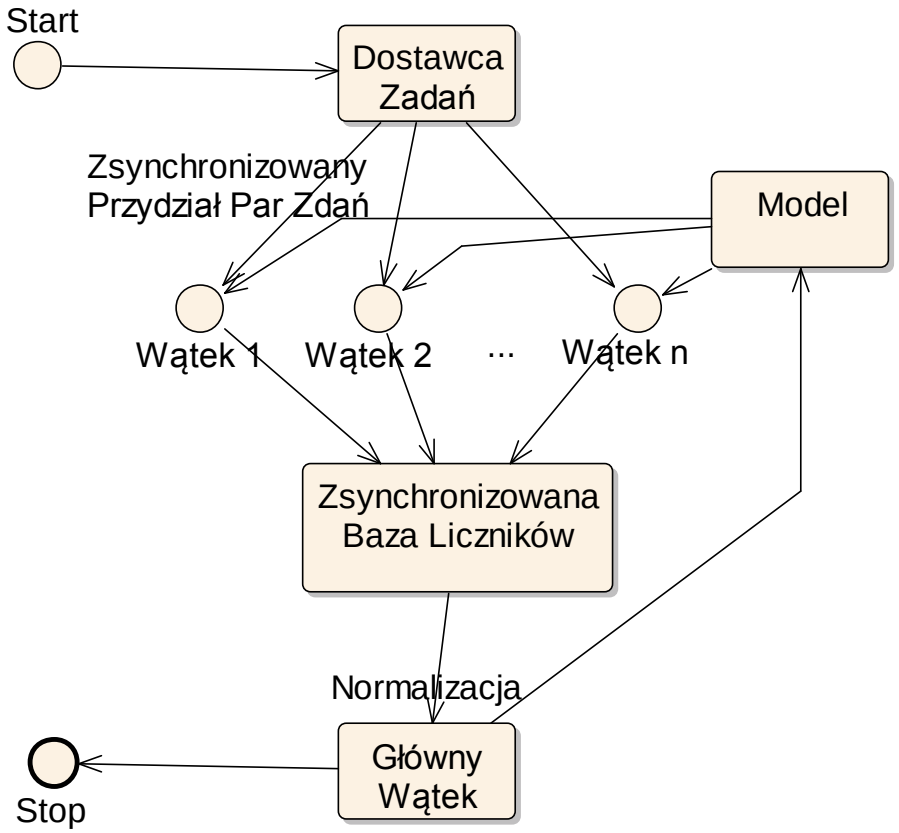
Rysunek 7. Algorytm działania programu Giza++ [19].

MGiza++

Giza++ cały proces dopasowywania wyrazów wykonuje szeregowo, co znacznie wydłuża czas obliczeń. Rozwiązano ten problem w narzędziu MGiza++. Wykorzystuje ono wiele rdzeni pojedynczej maszyny poprzez zastosowanie wielowątkowości. Algorytm działania przedstawiono na rysunku 8. Przy uruchomieniu programu definiuje się ilość wątków do obliczeń. Odpowiedni moduł wysyła do każdego z wątków parę zdań. Gdy jeden z wątków skończy obliczenia, pobiera kolejną parę zdań. Gdy cały korpus zostanie pobrany, następuje normalizacja wyników w głównym wątku, po czym algorytm przechodzi do kolejnej iteracji danego modelu lub uruchamia kolejny model. Po zakończeniu obliczeń, zwracane są wyniki do odpowiednich plików wyjściowych.

MGIZA++ jest omawiana w wielu pracach. Wykorzystuje ją między innymi T. Okita [22] w swoich obliczeniach, poprawiając współczynnik BLEU (określający jakość tłumaczenia automatycznego).

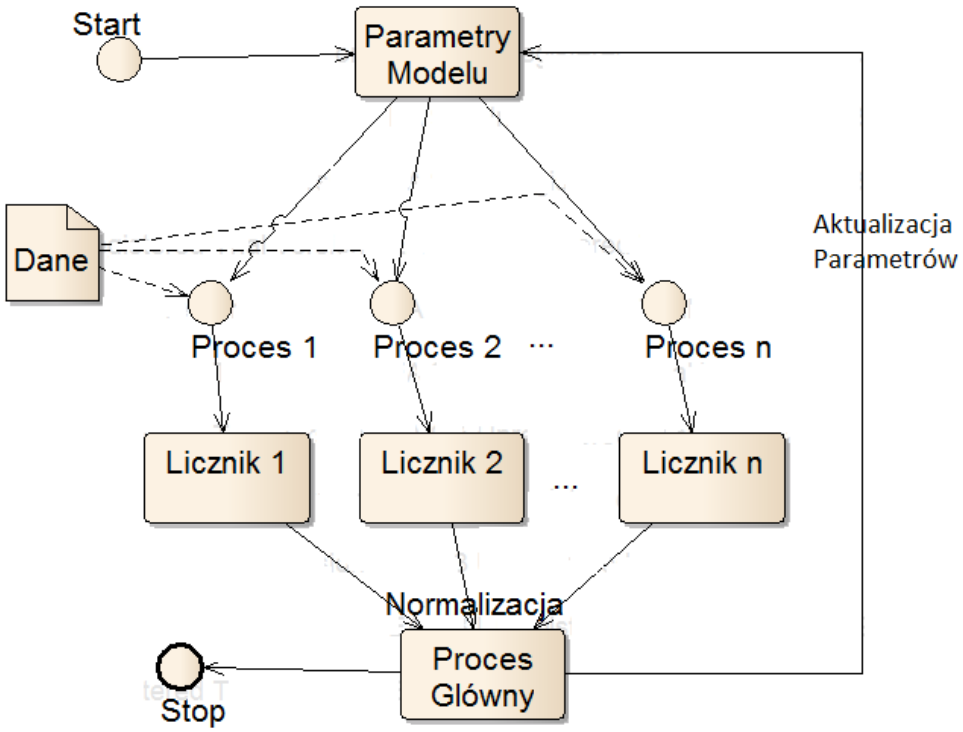
Podejście do problemu zastosowane w narzędziu MGiza++ znacznie skraca czas obliczeń, gdy mamy do czynienia z maszyną wielordzeniową. Natomiast, gdy dany algorytm MGiza++ uruchomimy z jednym wątkiem, to działa on w taki sam sposób jak algorytm Giza++.



Rysunek 8. Algorytm działania programu MGiza++ [19].

PGiza++

Inne podejście do problemu przedstawia narzędzie PGiza++. Wykonuje ono obliczenia na kilku maszynach, poprzez uruchomienie wielu procesów. Dzięki temu, na każdej maszynie można uruchomić ilość procesów odpowiadającą ilości rdzeni danej maszyny i czynność tą powielić na wielu maszynach. Sposób działania takiego algorytmu obrazuje rysunek 9. W tym przypadku do każdego procesu jest wysyłana część korpusu źródłowego. Za poprawność całego procesu odpowiada proces główny, który przydziela fragmenty korpusu dla każdego procesu i odbiera cząstkowe wyniki obliczeń. W procesie głównym następuje normalizacja wyników i przejście do kolejnej iteracji danego modelu lub uruchomienie kolejnego modelu. Po zakończeniu obliczeń proces główny zwraca wyniki obliczeń.



Rysunek 9. Algorytm działania programu PGiza++ [19].

Liczne testy wykazały, że narzędzie PGiza++ nie wykazuje znacznego skrócenia czasu obliczeń w porównaniu ze standardową Giza++. Jest kilka powodów takiego stanu:

- poszczególne procesy mogą nie zakończyć się jednocześnie, a kolejną iterację można rozpocząć dopiero po zakończeniu obliczeń we wszystkich procesach
- przesyłanie danych do poszczególnych wątków pochłania dodatkowy czas
- algorytm zawiera dodatkowe obliczenia: normalizację wraz z sumowaniem wyników obliczeń z poszczególnych procesów.

Aby zmniejszyć ilość przesyłanych danych, zastosowano współdzielony przez wszystkie maszyny zasób pamięci dyskowej. Jednak i to nie spowodowało znacznego przyśpieszenia czasu obliczeń.

Chaski

Rozszerzeniem koncepcji PGiza++ jest narzędzie Chaski [21]. Bazuje ono na algorytmie PGiza++ z pewnymi rozszerzeniami:

- zastosowano nowy system plików HDFS (ang. Hadoop Distributed File System) [23], nadający się znakomicie do systemów klastrowych,
- wykorzystano nowy sposób normalizacji wyników, spowodowany innym sposobem działania systemu plików HDFS,

- obliczenia, które zakończyły się niepowodzeniem, system uruchamia ponownie, dbając o integralność danych.

Zastosowanie tych zmian spowodowało znaczne przyspieszenie obliczeń. Od tej chwili narzędzie to może konkurować z MGiza++, jeśli chodzi o szybkość obliczeń.

Sprawdzono [21], [24], że przy identycznych danych wejściowych, wyniki zwracane przez każde z czterech wymienionych narzędzi są identyczne.

1.5.2. Narzędzia kompleksowe – LoonyBin, Moses

Narzędzia takie Giza++, MGiza++ potrzebują specyficznego formatu danych wejściowych. W tym celu należy wykonać szereg komend, aby z korpusu wejściowego otrzymać dane wejściowe do danego narzędzia. Aby ten proces zautomatyzować, a także połączyć ze sobą szereg narzędzi przydatnych w statystycznym tłumaczeniu automatycznym powstały aplikacje LoonyBin [25], [26] i Moses [27].

O ile Moses jest narzędziem znanym i szeroko stosowanym, o tyle LoonyBin powstało w 2010 roku i jest narzędziem mało popularnym.

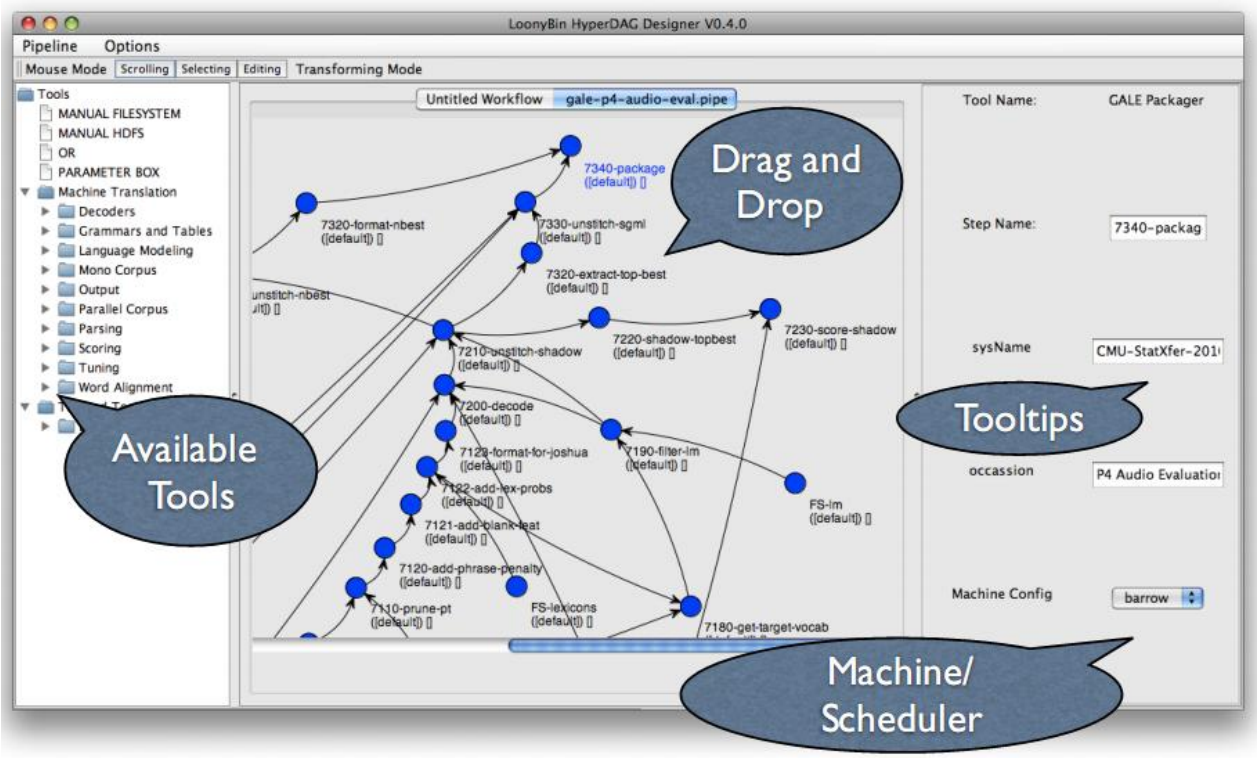
Moses to system statystycznego tłumaczenia automatycznego. W sposób kompleksowy podchodzi do zadania tłumaczenia automatycznego. Jako parametry wejściowe przyjmuje korpus dwujęzyczny i plik z ustawieniami wejściowymi programu. Umożliwia wykonanie obliczeń w dwóch kierunkach – od korpusu źródłowego do docelowego i od docelowego do źródłowego. Otrzymane w ten sposób dwukierunkowe dopasowania słów system automatycznie łączy poprzez dostępne algorytmy symetryzacji. Zamiast Gizy++, można podłączyć do systemu MGizę++ lub inny program zwracający te same pliki wynikowe.

Natomiast LoonyBin (rysunek 10.) jest narzędziem napisanym w języku Java z intuicyjnym interfejsem graficznym. Umożliwia wykonywanie obliczeń na wielu maszynach. Główny program odpowiada za wykrywanie dostępnych narzędzi i zdalnych maszyn w systemie. Sprawdza poprawność plików wynikowych z każdego uruchomionego procesu. Zawiera system równoważenia przydzielonych zadań (ang. *load balancing*), a także wiele przydatnych narzędzi:

- dopasowania wyrazów: Moses, MGiza++, Chaski,
- sprawdzania jakości dopasowania i tłumaczenia: BLEU, NIST [28], Meteor [29], TER [30],
- wiele innych np.: SAMT (składniowe tłumaczenie maszynowe) [31], MEMT (system tłumaczenia automatycznego) [32].

W łatwy sposób umożliwia dodawanie nowych aplikacji.

Na rysunku 10. przedstawiono interfejs graficzny narzędzia LoonyBin



Rysunek 10. Interfejs graficzny narzędzia LoonyBin [33]

Rozdział 2

Technika symetryzacji dopasowania tekstu jako sposób poprawy jakości dopasowania wyrazów

2.1. Wprowadzenie

Definicja 12. Symetryzacja dopasowania wyrazów – etap procesu tworzenia macierzy powiązań, w którym następuje powiązanie informacji z dwóch kierunków obliczeń dopasowań metodami statystycznymi.

Dzięki zastosowaniu tej operacji uzyskuje się połączenia 1 do n i n do 1 między wyrazami, gdzie n>=1. Na tej podstawie tworzy się dopasowania wynikowe.

Zastosowanie symetryzacji dopasowania wyrazów metodami statystycznymi w korpusach dwujęzycznych pozwala poprawić jakość tłumaczeń wynikowych. Poprawa następuje w dwóch aspektach:

- zwiększenie prawdopodobieństwa dopasowania odpowiadających sobie wyrazów, gdyż proces połączenia następuje w obu kierunkach obliczeń,
- powiązanie wielowyrazowych fragmentów tekstów.

Ten drugi aspekt rozbudowuje otrzymaną macierz powiązań, zwiększając dostępne opcje tłumaczeń wyrazów.

2.2. Opis technik symetryzacji przy dopasowaniu tekstu

2.2.1. Symetryzacja wyników końcowych

Symetryzacja wyników końcowych jest mechanizmem szeroko stosowanym. Istnieją dostępne narzędzia służące do tego celu. Jednym z nich jest MOSES [27] – narzędzie przeznaczone do tworzenia statystycznych translatorów automatycznych, skonstruowane w sposób modułowy, przy czym jeden z modułów służy do symetryzacji. Wywołanie tego modułu po procesie obliczeń dopasowania wyrazów w obu kierunkach uruchamia symetryzację. Symetryzacja może być wykonana na kilka sposobów. W zależności od algorytmu, jaki zostanie użyty, można uzyskać albo znaczne polepszenie trafności

dopasowań, albo jakości tłumaczenia końcowego, bądź wyważenie pomiędzy tymi dwoma aspektami.

2.2.2. Rodzaje symetryzacji – iloczyn, suma mnogościowa, ich kompilacje

Definicja 13. Plikiem wynikowym dopasowania wyrazów dla korpusu dwujęzycznego nazywamy listę macierzy dopasowania wyrazów, które stanowią reprezentację powiązań między wyrazami w zdaniach będących swoim wzajemnym tłumaczeniem.

W przykładzie 10. zaprezentowano fragment pliku wynikowego zawierającego trzy pary zdań dla korpusu angielsko-francuskiego. W pliku wynikowym dopasowania dla każdej pary zdań są tworzone trzy wiersze zawierające odpowiednio:

- informację o ilości wyrazów w zdaniu źródłowym i docelowym, oraz prawdopodobieństwu poprawnego dopasowania zdań,
- zdanie źródłowe,
- zdanie docelowe z przypisanymi identyfikatorami wyrazów ze zdania źródłowego.

Przykład 10. Fragment macierzy dopasowań korpusu francusko-angielskiego w postaci pliku wynikowego.

 # Sentence pair (1) source length 2 target length 2 alignment score : 0.244269
<CHAPTER ID=1>
NULL ({ }) <CHAPTER ({ 1 }) ID=1> ({ 2 })
 # Sentence pair (2) source length 4 target length 4 alignment score : 0.0015392
Reprise de la session
NULL ({ }) Resumption ({ 1 }) of ({ 2 }) the ({ 3 }) session ({ 4 })
 # Sentence pair (3) source length 3 target length 4 alignment score : 0.0103554
<SPEAKER ID=1 NAME="La Présidente">
NULL ({ }) <SPEAKER ({ 1 }) ID=1 ({ 2 }) NAME="President"> ({ 3 4 })

Załóżmy, że celem jest dopasowanie wyrazów w korpusie polsko-angielskim. W pierwszym kroku dokonujemy dopasowania w kierunku polsko-angielskim, w wyniku którego otrzymujemy pierwszy plik wynikowy. W drugim kroku operację powtarzamy w kierunku angielsko-polskim, otrzymując drugi plik wynikowy. Proces symetryzacji polega w takim przypadku na odpowiednim zagregowaniu wyników zapisanych w obu plikach.

Symetryzacja dopasowań wyrazów może być wykonana na wiele sposobów. Rozróżniamy dwa podstawowe typy symetryzacji:

- symetryzację według iloczynu mnogościowego,

- symetryzację według sumy mnogościowej.

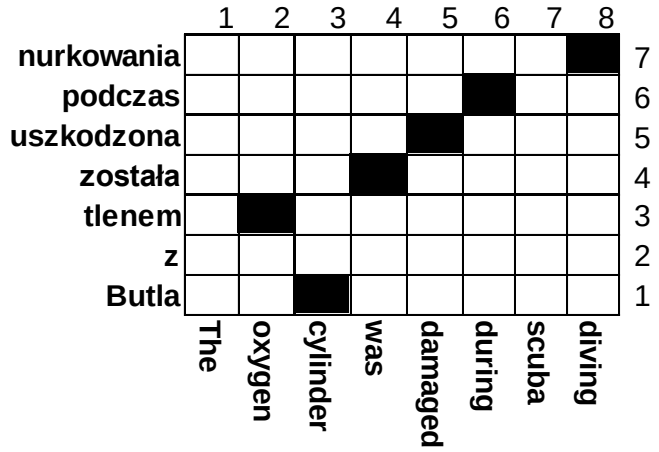
Definicja 14. Symetryzacją według iloczynu mnogościowego nazywamy symetryzację, w której prawdopodobieństwa powiązań rozpatrywane są wyłącznie dla par wyrazów (fraz) występujących w obu plikach wynikowych. W wyniku takiej symetryzacji otrzymujemy macierz A powstałą w następujący sposób: $A = A_1 \cap A_2$, gdzie A_1 i A_2 są macierzami dopasowań jednokierunkowych.

Symetryzację według iloczynu mnogościowego zobrazujemy na przykładzie, w którym analizowany korpus dwujęzyczny składa się z jednej pary zdań: zdania polskiego i odpowiadającego mu zdania angielskiego (oba pliki wynikowe składają się z jednej macierzy dopasowania wyrazów).

Przykład 11. Symetryzacja według iloczynu mnogościowego

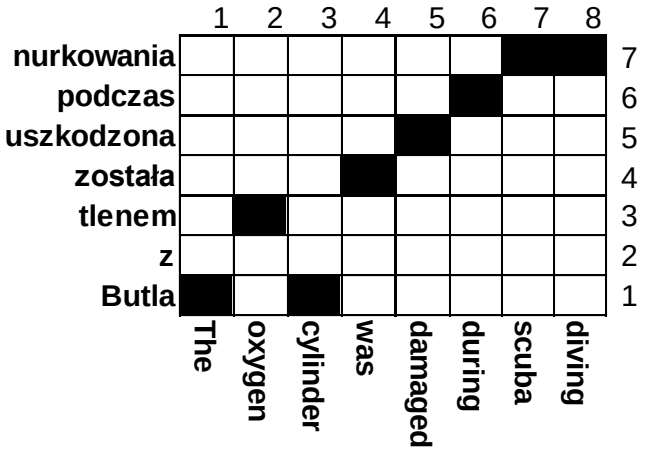
Zdanie w języku polskim: *Butla z tlenem została uszkodzona podczas nurkowania.*

Zdanie w języku angielskim: *The oxygen cylinder was damaged during scuba diving.*



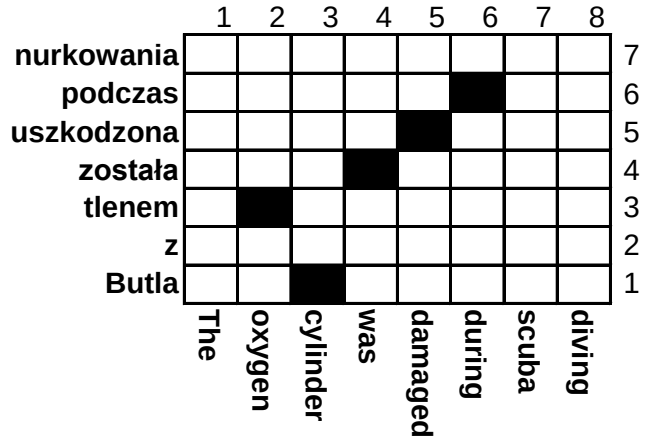
Rysunek 11. Dopasowanie w kierunku: język polski do język angielski.

Rysunek 11. przedstawia dopasowanie wyrazów w kierunku polsko-angielskim. W takim przypadku każdy wyraz angielski może być powiązany z n wyrazami polskimi. Ponadto każdemu wyrazowi polskiemu odpowiada maksymalnie jeden wyraz angielski. Wyrazy *„The”* i *„scuba”* nie mają powiązania z żadnym wyrazem.



Rysunek 12. Dopasowanie w kierunku: język angielski do język polski.

Rysunek 12. przedstawia dopasowanie wyrazów w kierunku angielsko-polskim. W tym przypadku każdy wyraz polski może być powiązany z n wyrazami angielskimi. Ponadto każdemu wyrazowi angielskiemu odpowiada maksymalnie jeden wyraz polski. Wyrazowi „*nurkowania*" odpowiadają wyrazy *„scuba"* i *„diving"*, natomiast wyrazy *„z"* nie ma swojego odpowiednika w zdaniu angielskim.



Rysunek 13. Symetryzacja według iloczynu mnogościowego dla dopasowania polsko-angielskiego.

Macierz na rysunku 13. jest wynikiem symetryzacji według iloczynu mnogościowego. Jak widzimy, występują w niej tylko powiązania występujące zarówno w dopasowaniu polsko-angielskim jak i angielsko-polskim.

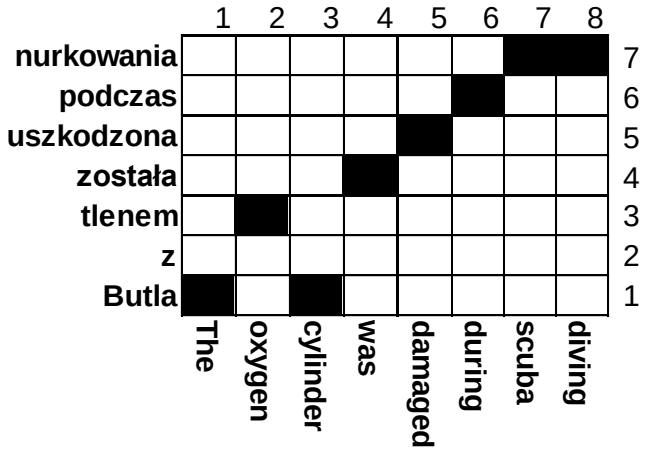
Definicja 15. Symetryzacją według sumy mnogościowej nazywamy symetryzację, w której prawdopodobieństwa powiązań rozpatrywane są dla par wyrazów (fraz) występujących w dowolnym pliku wynikowym. W wyniku takiej symetryzacji otrzymujemy macierz A ,

powstałą w następujący sposób $A = A_1 \cup A_2$, gdzie A_1 i A_2 są macierzami dopasowań jednokierunkowych.

Przykład 12. Symetryzacja według sumy mnogościowej

Zdanie w języku polskim: *Butla z tlenem została uszkodzona podczas nurkowania.*

Zdanie w języku angielskim: *The oxygen cylinder was damaged during scuba diving.*



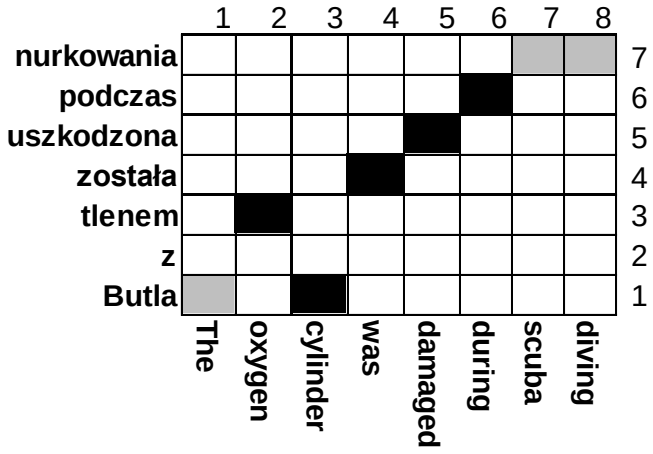
Rysunek 14. Symetryzacja według sumy mnogościowej dla dopasowania polsko-angielskiego.

Rysunek 14. przedstawia macierz dopasowania będącą wynikiem symetryzacji według sumy mnogościowej. W omawianym przypadku wyrazowi „*nurkowania*" przyporządkowano sentencję „*scuba diving*".

Poza przedstawionymi powyżej dwoma bazowymi metodami symetryzacji dopasowań występuje także szereg kompilacji tych dwóch metod. Jedną z najpopularniejszych jest metoda *refined* [12]. W tej metodzie wykonuje się symetryzację według iloczynu mnogościowego, a następnie dodaje się dopasowania (i, j) występujące w macierzy A_1, bądź A_2, jeżeli ani wyraz z macierzy A_1, ani wyraz z A_2 nie ma dopasowania w macierzy A lub spełnione są zarazem dwa warunki:

- dopasowanie (i, j) ma poziomych $(i-1, j)$, $(i+1, j)$ lub pionowych $(i, j-1)$, $(i, j+1)$ sąsiadów, którzy są w macierzy A ,
- zbiór $A \cup \{(i, j)\}$ nie zawiera dopasowań zarówno z poziomymi i pionowymi sąsiadami.

Aby zobrazować działanie metody *refined*, użyjemy przykładu prezentowanego na rysunku 13., przedstawiającego symetryzację według iloczynu mnogościowego. W ten sposób zrealizowano pierwszy etap metody *refined*. Teraz należy znaleźć odpowiedniki dla wyrazów, które nie mają jeszcze dopasowania. W naszym przypadku są to wyrazy: *„The”, „scuba”* i *„diving”*. Dopasowania tych wyrazów występują w macierzach z rysunku 11. i rysunku 12., więc dodajemy je do wynikowego dopasowania. W ten sposób otrzymujemy dopasowanie metodą *refined* zaprezentowane na rysunku 15.



Rysunek 15. Symetryzacja według metody refined dla dopasowania polsko-angielskiego.

2.3. Miary oceny jakości dopasowania.

Do porównania trafności dopasowań uzyskanych różnymi metodami, stosuje się następujące miary:

- precyzję,
- pokrycie,
- współczynnik błędu dopasowań AER (ang. Alignment Error Rate).

Miary te zostaną omówione w podrozdziale 2.3.1. Ponadto, jakość dopasowania można ocenić w sposób pośredni – ewaluując wyniki tłumaczenia automatycznego uzyskanego na bazie dopasowania. Przykładową miarę jakości tłumaczenia przedstawiono w podrozdziale 2.3.2

2.3.1. Miary oceny bezpośredniej dopasowania

Miary jakości dopasowania określa się w odniesieniu do dopasowania referencyjnego, uznanego za wzorcowe. W dopasowaniu wzorcowym wyróżnia się dwa typy powiązań:

- powiązania pewne – poprawne „z całą pewnością”,
- powiązania dopuszczalne – pojawiające się w sytuacjach niejednoznaczności.

Definicja 16. Dopasowanie referencyjne to dopasowanie, w którym wszystkie wskazane powiązania są albo pewne, albo dopuszczalne, a ponadto obejmuje cały zbiór takich powiązań.

Dopasowanie referencyjne tworzone jest z reguły przez człowieka.

Definicja 17. Precyzja (ang. precision) – w ogólności jest to stosunek liczby poprawnych wskazań zwróconych przez algorytm, do liczby wszystkich zwróconych wskazań. Miarę tą można zastosować do określenia dokładności danego dopasowania wyrazów w korpusie dwujęzycznym za pomocą następującego wzoru:

$$\mathrm{Pr}ecision = \frac{|A \cap P|}{|A|},$$

gdzie P oznacza powiązania pewne lub dopuszczalne w dopasowaniu referencyjnym, a A zawiera wszystkie powiązania wskazane przez algorytm.

Definicja 18. Pokrycie (ang. recall) – to stosunek liczby wskazań zwróconych przez algorytm, do liczby wszystkich przypadków, które algorytm powinien wskazać. Miarę tą można zastosować do określenia pokrycia danego dopasowania wyrazów w korpusie dwujęzycznym za pomocą następującego wzoru:

$$\mathrm{Re}call = \frac{|A \cap S|}{|S|},$$

gdzie S jest zbiorem powiązań pewnych w dopasowaniu referencyjnym.

Współczynnik błędu dopasowania został zaproponowany przez Och'a i Ney'a [12] w 2003 roku. Od tego czasu stosuje się go w wielu pracach porównujących otrzymane wyniki doświadczeń, bądź prezentujących nowe wyniki [34].

Mechanizm wyliczania współczynnika błędu dopasowania zwraca procentową trafność dopasowania wyrazów w korpusie dwujęzycznym.

Definicja 19. Współczynnik błędu dopasowania (ang. Alignment Error Rate) - współczynnik określający trafność dopasowania poszczególnych wyrazów w pliku wynikowym dopasowania.

Na podstawie tłumaczenia referencyjnego system stwierdza, czy dane dopasowanie wyrazów jest prawidłowe. Porównując kolejne zdania dla całego korpusu referencyjnego, powstaje współczynnik AER, odzwierciedlający trafność powiązań wyrazów.

Współczynnik AER przedstawiający ilość błędów dopasowań oblicza się z następującego wzoru:

$$\mathrm{AER} = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}, \tag{33}$$

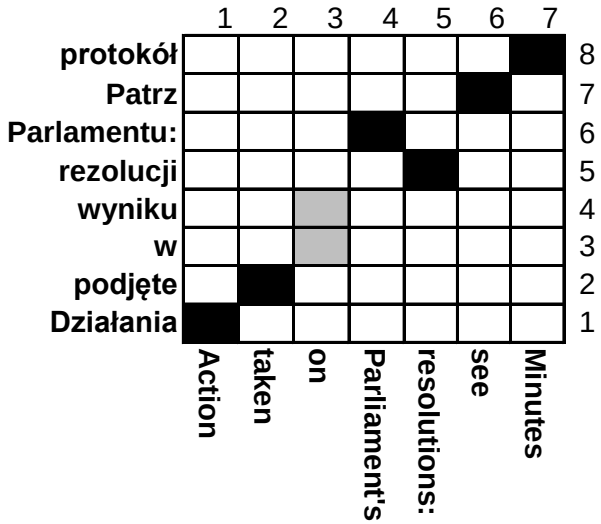
gdzie A, P i S są zdefiniowane jak wyżej

Przykład 13. Etapy wyznaczania współczynnika błędu dopasowania wyrazów dla przykładowej pary sentencji korpusu polsko-angielskiego.

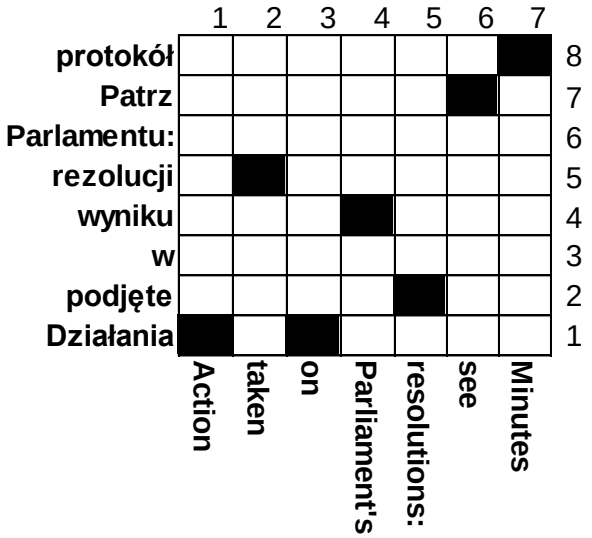
Załóżmy, że dla pary zdań w tłumaczeniu polsko-angielskim:

 # Sentence pair (1) source length 8 target length 7 alignment score : 7.04454e-05
Action taken on Parliament's resolutions: see Minutes
NULL ({ }) Działania ({ 1 3 }) podjęte ({ 5 }) w ({ }) wyniku ({ 4 }) rezolucji ({ 2 }) Parlamentu: ({ }) Patrz ({ 6 }) protokół ({ 7 }),

mamy następujący referencyjny zbiór dopasowań pewnych (kolor czarny) i dopuszczalnych (kolor szary)



Natomiast oceniony zbiór dopasowań zdefiniowany za pomocą pliku wynikowego dopasowania w formie macierzy dopasowań ma postać:



Na tej podstawie możemy zdefiniować precyzję, pokrycie i współczynnik błędu dopasowania:

- Precyzja = 3/7=0,42,
- Pokrycie = 3/6 = 0,5,
- AER = 1 – (3+3)/(7+6) = 1 – 6/13 = 1 – 0,46 = 0,54.

Podczas porównania metod symetryzacji obliczeń dwukierunkowych wykorzystuje się pojęcia precyzji (definicja 17.) i pokrycia (definicja 18.) odzwierciedlające użyteczność poszczególnych metod. Metoda iloczynu dopasowań charakteryzuje się wyższą precyzją i mniejszym pokryciem, natomiast metoda sumy dopasowań skutkuje większym pokryciem i mniejszą precyzją niż dopasowania jednokierunkowe. W wynikowych danych dopasowania wyrazów oczekujemy wysokiej precyzji i dużego pokrycia, z czego ważniejszą cechą jest duże pokrycie [35]. Metoda *refined*, wykorzystująca zalety obu metod bazowych daje najlepsze rezultaty pod względem trafności dopasowań (najmniejszy współczynnik błędu).

2.3.2. Opis miary BLEU jako narzędzia opisującego jakość tłumaczenia

Jedną z miar jakości tłumaczenia jest miara BLEU (ang. Bilingual Evaluation Understudy) [36]. Jakość tłumaczenia jest w tej mierze oceniana jako korelacja pomiędzy danymi wyjściowymi z danego systemu a tłumaczeniem referencyjnym – będącym wynikiem pracy tłumacza ludzkiego. Wyniki porównania tłumaczeń poszczególnych zdań są uśredniane, aby otrzymać obraz całości tłumaczenia. Miara BLEU jest popularnym narzędziem do

porównania poszczególnych mechanizmów tłumaczeń. Mimo kilku wad opisanych w poniższych paragrafach pozostaje jednym z wyznaczników dobrej jakości tłumaczenia.

Warto wspomnieć, że miara BLEU ma stosunkowo wysoką korelację z ludzkim osądem tłumaczenia automatycznego [37], [38].

Aby w pełni zrozumieć zasadę obliczania miary BLEU, należy wprowadzić pojęcie n-gramu.

Definicja 20. N-gram – jest to podciąg składający się z n wyrazów danego ciągu. W zależności od długości podciągu n-gram przyjmuje różne nazewnictwo:

- unigram – n-gram o długości 1,
- bigram – n-gram o długości 2,
- trigram – n-gram o długości3,
- dłuższe podciągi nazywamy po prostu n-gramami.

W algorytmie jakości dopasowania wyrazów miarą BLEU dla danego zdania sprawdza się czy n-gramy ze zdania przetłumaczonego automatycznie występują w przynajmniej jednym z tłumaczeń referencyjnych.

Stosuje się pojęcia precyzji oraz pokrycia w sposób analogiczny do oceny trafności dopasowań (nie rozróżnia się przypadków dopuszczalnych i pewnych):

$$\mathrm{Pr}ecision = \frac{|A \cap P|}{|A|}, \tag{34}$$

$$\mathrm{Re}call = \frac{|A \cap P|}{P}, \tag{35}$$

gdzie P jest zbiorem n-gramów występujących tłumaczeniu referencyjnym, a A jest zbiorem n-gramów występujących w tłumaczeniu ocenianym.

Podejście to doprowadza niekiedy do nieoczekiwanych wyników, co obrazuje przykład 14.

Przykład 14. Algorytm jakości dopasowania wyrazów miarą BLEU

Załóżmy, że w tłumaczeniu wygenerowano powtórzenia tego samego wyrazu. Skrajny przypadek takiej sytuacji obrazuje poniższa tabela:

Oceniane tłumaczenie	issue	issue	issue	issue	issue	issue	issue	issue	issue	issue
Referencja 1	the	issue	of	safety	at	sea	is	of	vital	importance
Referencja 2	the	issue	of	safety	at	sea	is	very	important	

Tabela 6. Przykład tłumaczenia z niepożądanymi powtórzeniami

W przedstawionym przykładzie, precyzja wyliczona dla unigramów wynosi:

$$P = \frac{m}{w_t} = \frac{10}{10} = 1, \tag{36}$$

gdzie m jest liczbą wyrazów kandydata znalezionych w tekście referencji, a w liczbą wyrazów referencji.

W danej metryce wszystkie dziesięć słów tłumaczenia kandydującego znajduje się w tłumaczeniu referencyjnym. Precyzja wynosi 1, pomimo że tłumaczenie kandydata nie zachowuje treści z żadnego tłumaczenia referencyjnego. Dlatego też stosuje się pewną modyfikację: dla każdego wyrazu w tłumaczeniu kandydującym, bierze się pod uwagę maksymalną całkowitą liczbę wystąpień danego wyrazu w tłumaczeniu referencyjnym. W powyższym przykładzie wyraz „issue" pojawia się raz w referencji 1 i raz w referencji 2. To dlatego jako maksymalną liczba wystąpień tego wyrazu przyjmuje się 1.

W kolejnym kroku obliczania miary BLEU dla każdego wyrazu w tłumaczeniu kandydującym, porównuje się liczbę powtarzających się słów: m_w i maksymalną liczbę powtórzeń danego wyrazu we wszystkich tłumaczeniach referencyjnych: $m_{\max}$, a następnie wybiera się z tego zbioru wartość niższą:

$$Count_{clip} = \min(m_w, m_{\max}) \tag{37}$$

W tym przypadku liczba słów „issue" w tłumaczeniu kandydującym wynosi 10, więc $m_w = 10$, a maksymalną liczbą referencyjną jest 1, czyli $m_{\max} = 1$. Z tego wynika, że $Count_{clip} = 1$. Następnie wartość $Count_{clip}$ jest dzielona przez całkowitą liczbę słów w tłumaczeniu kandydującym. Na podstawie powyższego przykładu, otrzymujemy następującą wartość precyzji:

$$P = \frac{Count_{clip}}{m} = \frac{1}{10} \tag{38}$$

Wykonując obliczenia dla n-gramów różnej długości, otrzymujemy różne wyniki. Z przeprowadzonych doświadczeń wynika, że wartość n mająca „największą korelację z jednojęzycznym osądem ludzkim" wynosi cztery.

Końcowym etapem tworzenia miary BLEU dla całego korpusu, jest łączenie wartości precyzji dla segmentów. Wykorzystuje się średnią geometryczną wartości precyzji pomnożoną przez karę zwięzłości (ang. brevity penalty) w celu uniknięcia sytuacji, w której krótkie teksty kandydujące otrzymują zbyt wysoki wynik. Niech r będzie całkowitą

długością korpusu referencyjnego (liczbą wyrazów w korpusie), a c całkowitą długością korpusu kandydata. Jeśli $c \le r$ to do obliczenia kary zwięzłości wykorzystuje się następujący wzór: $BP = e^{(1-\frac{r}{c})}$. W przypadku wielu różnych zdań referencyjnych, r to suma długości zdań, których długości są najbliższe do długości zdań kandydujących.

Definicja 21. Miara BLEU (ang. Bilingual Evaluation Understudy) – miara jakości tłumaczenia określana jako korelacja pomiędzy danymi wyjściowymi z danego systemu tłumaczącego a tekstem referencyjnym. Miara ta jest określona wzorem:

$$BLEU = BP \cdot e^{(\sum_{n=1}^{N} w_n \log p_n)}, \tag{39}$$

gdzie BP oznacza karę zwięzłości określoną wzorem $BP = \begin{cases} 1 & \text{jesli c} > \text{r} \\ e^{(1-\frac{r}{c})} & \text{jesli c} \le \text{r} \end{cases}$,

w_n oznacza liczbę wyrazów referencji, a p_n oznacza precyzję wyliczoną dla n-gramów.

Przykład 15. Obliczenie miary BLEU dla dwóch zdań języka polsko-angielskiego.

Załóżmy, że na wejściu systemu tłumaczącego mamy następujący tekst:

W wyniku podjętych działań wojennych zginęło kilka tysięcy osób. Rząd nie poczuwa się do odpowiedzialności za tą tragedię.

Natomiast na wyjściu otrzymujemy wynik w postaci:

As a result of military operations killed several thousand people. The government feel no responsibility for this tragedy.

Tłumaczenie referencyjne (ręczne) dla powyższego tekstu ma następującą postać:

As a result of military operations several thousand people were killed. The government feel no responsibility for this tragedy.

Obliczmy precyzję dla unigramów (definicja 20.) dla każdego zdania porównując tłumaczenie wynikowe z tłumaczeniem referencyjnym:

- P_1 = 1/10 + 1/10+ 1/10 + 1/10 + 1/10+ 1/10+ 1/10+ 1/10+ 1/10+ 1/10 = 1
- P_2 = 1/8 +1/8 + 1/8 + 1/8 + 1/8 + 1/8 + 1/8 + 1/8 = 1

Końcowym etapem obliczenia miary BLEU jest powiązanie wszystkich wyników. Wykorzystuje się w tym celu średnią geometryczną wartości precyzji pomnożoną przez karę zwięzłości:

$$BLEU = e^{(1-20/18)} \cdot e^{(11 \cdot \log 1 + 8 \cdot \log 1)} = e^{-\frac{2}{18}} \cdot e^{0} \approx \frac{1}{e^{0,1111}} \approx \frac{1}{1,1175} \approx 0,8948$$

Miara BLEU w wielu przypadkach okazuje się dobrze korelować z ludzkim osądem, dzięki temu stała się punktem odniesienia dla oceny każdej nowej i zmodyfikowanej metody. W kilku przypadkach stwierdzono, że ocena ludzka różniła się od oceny BLEU [38], [40]. Mimo to metoda ta jest nadal szeroko stosowana i uzyskuje wiele pozytywnych opinii.

2.4. Wpływ symetryzacji na wartość współczynników AER i BLEU

Symetryzacja dwukierunkowych obliczeń dopasowania wyrazów metodami statystycznymi powoduje polepszenie zarówno współczynnika AER jak i miary BLEU. Dzięki wykonaniu takiej symetryzacji system otrzymuje znacznie więcej informacji o prawdopodobieństwach dopasowania poszczególnych wyrazów, przez co może z większą dokładnością dopasować wyrazy.

W zależności od zastosowanego algorytmu symetryzacji możemy uzyskać większą trafność dopasowania (wzrost AER), bądź wyższą jakość tłumaczenia końcowego (wzrost BLEU).

Rozdział 3

Opis rozwiązania autorskiego

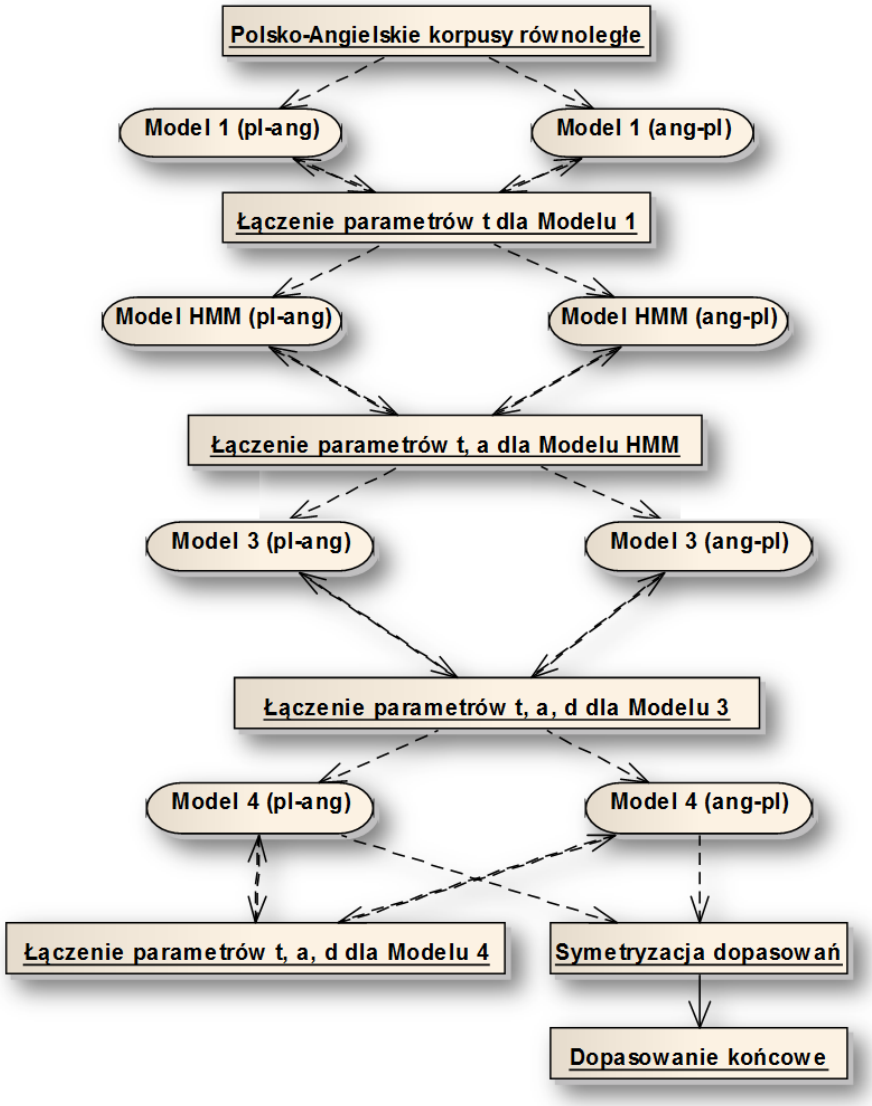
3.1. Zastosowanie symetryzacji wewnątrz iteracji – operacje na plikach poszczególnych modeli

Podejście wykorzystujące symetryzację wewnątrz procesu dopasowania wyrazów wymaga znacznej ingerencji w modele statystyczne, co powoduje duże modyfikacje stosowanych narzędzi. Jak dotąd, nie przedstawiono prac proponujących podejście do zagadnienia poprzez modyfikację modeli statystycznych. W mojej pracy pragnę przedstawić wyniki, uzyskane właśnie poprzez odpowiednią modyfikację algorytmów dopasowania wyrazów. Symetryzacja obliczeń następuje po każdej iteracji (w zależności od konfiguracji) konkretnego modelu statystycznego z obu kierunków obliczeń. To wymaga zastosowania wielowątkowości, aby obliczenia w jednym czasie były wykonywane w dwóch kierunkach. Po każdej iteracji modelu statystycznego następuje powiązanie wyników z obu kierunków obliczeń z zastosowaniem odpowiednich przekształceń. Następnie, tak powstałe tablice są przekazywane do obu kierunków obliczeń i następuje kolejna iteracja.

Aby wykonać symetryzację dopasowań wyrazów, należy wykonać po każdej iteracji powiązanie odpowiednich plików wyjściowych z dwóch kierunków obliczeń. W zależności od aktualnego modelu statystycznego liczba plików wymagających powiązania zmienia się, podobnie jak ich wielkość. Zawartość poszczególnych plików wraz z modelami statystycznymi przedstawiono w podrozdziale 3.1.1.

3.1.1. Opis zawartości plików wynikowych modeli statystycznych

Pełen proces symetryzacji obliczeń dwukierunkowych przedstawia rysunek 16. Po każdej iteracji modelu 1. wykonujemy symetryzację plików *t* (zawierających prawdopodobieństwa tłumaczenia wyrazów). W modelu HMM wykonujemy symetryzację plików *t* i *a* (zawierających prawdopodobieństwa dopasowania pozycji wyrazów). Natomiast w modelu 3. i 4. dochodzi dodatkowo symetryzacja plików *d* (zawierających prawdopodobieństwa zniekształcenia wyrazów na odpowiednich pozycjach). Po zakończeniu obliczeń następuje symetryzacja końcowa wykonywana jedną z dostępnych metod (patrz rozdział 2.2).



Rysunek 16. Schemat działania symetryzacji po każdej iteracji.

Kolejne pliki *t* mają strukturę przedstawioną w tabeli 7. Zawierają one prawdopodobieństwa tłumaczenia wyrazów w danym korpusie dwujęzycznym w jednym z kierunków. Liczba w pierwszej kolumnie oznacza unikalny identyfikator wyrazu z korpusu docelowego, natomiast druga kolumna zawiera identyfikator wyrazu z korpusu źródłowego. Ostatnia kolumna zawiera prawdopodobieństwo dopasowania dwóch wyrazów.

```
5 5 0.974639
5 7 1.18046e-06
5 15 0.000361147
5 28 0.000914295
5 51 0.000450937
5 52 1.5565e-07
5 95 0.000456132
5 98 3.06115e-06
5 174 0.000390813
5 231 0.00100435
5 252 6.09241e-07
5 491 0.000454874
5 562 2.40404e-06
5 574 5.28884e-07
5 590 3.05958e-05
5 709 0.000909315
5 733 0.000423606
5 883 0.000913864
5 1016 0.000457147
5 1075 0.000492484
5 1149 0.000457255
5 1153 0.000457142
5 1282 0.000457142
```

Tabela 7. Struktura pliku t – prawdopodobieństwa tłumaczenia wyrazów, będąca wynikiem cząstkowym procesu dopasowania wyrazów metodami statystycznymi.

Struktura plików *a* została przedstawiona w tabeli 8. Pierwsze dwie kolumny oznaczają pozycję docelowego (j) i źródłowego wyrazu (i). Kolejne dwie zawierają długość źródłowej (l) i docelowej sentencji (m). Ostatnia kolumna przedstawia prawdopodobieństwo, że wyraz źródłowy na pozycji i ma swoje tłumaczenie na pozycji j w danej parze zdań długości l i m (p(i|j, l, m)).

0 34 8 100 0.351737
1 34 8 100 0.0588222
2 34 8 100 0.0582356
3 34 8 100 0.117621
4 34 8 100 6.68667e-05
5 34 8 100 0.0605918
6 34 8 100 0.176464
7 34 8 100 0.176461
0 35 8 100 0.52527
1 35 8 100 9.67917e-07
2 35 8 100 0.0624452
4 35 8 100 0.160417
5 35 8 100 0.00198577
6 35 8 100 4.44565e-06
7 35 8 100 0.124951
8 35 8 100 0.124926
0 36 8 100 0.379515
2 36 8 100 0.122281
5 36 8 100 0.0870917
6 36 8 100 0.140928
7 36 8 100 0.14286
8 36 8 100 0.127324

Tabela 8. Struktura pliku a – prawdopodobieństwa dopasowania pozycji wyrazu z pozycji i z wyrazem z pozycji j, będąca wynikiem cząstkowym procesu dopasowania wyrazów metodami statystycznymi.

Struktura plików *d* zawiera prawdopodobieństwo zniekształcenia, zmienione zostały jedynie pozycje słowa źródłowego i docelowego. W związku z tym pierwsza kolumna oznacza pozycję wyrazu źródłowego, natomiast druga kolumna zawiera pozycję wyrazu docelowego. Struktura ta została przedstawiona w tabeli 9.

```
1 2 100 8 0.0329446
2 2 100 8 0.545829
3 2 100 8 0.198938
4 2 100 8 0.0799517
5 2 100 8 0.0584022
6 2 100 8 0.0375255
7 2 100 8 0.036435
8 2 100 8 0.00997313
1 3 100 8 0.0216004
2 3 100 8 0.145818
3 3 100 8 0.331545
4 3 100 8 0.260529
5 3 100 8 0.0986671
6 3 100 8 0.0674201
7 3 100 8 0.0523016
8 3 100 8 0.0221183
1 4 100 8 0.0154062
2 4 100 8 0.06149
3 4 100 8 0.115336
4 4 100 8 0.294728
5 4 100 8 0.237137
6 4 100 8 0.113299
7 4 100 8 0.127221
8 4 100 8 0.0353819
1 5 100 8 0.0171707
2 5 100 8 0.0309011
3 5 100 8 0.045333
4 5 100 8 0.110997
5 5 100 8 0.288198
6 5 100 8 0.227277
7 5 100 8 0.177364
8 5 100 8 0.10276
1 6 100 8 0.00872781
2 6 100 8 0.0203154
3 6 100 8 0.0283079
4 6 100 8 0.0522318
5 6 100 8 0.105635
```

Tabela 9. Struktura pliku d – prawdopodobieństwa zniekształcenia wyrazu z pozycji j z wyrazem z pozycji i, będąca wynikiem cząstkowym procesu dopasowania wyrazów.

3.1.2. Opis algorytmów symetryzacji dla poszczególnych modeli statystycznych

Definicja 22. Algorytm symetryzacji nazywamy **poprawnym,** jeżeli zwraca macierz dopasowań, będącą złożeniem dwóch macierzy jednokierunkowych zgodnie

z uwarunkowaniami, jakie zostały nałożone na ten typ symetryzacji (np. symetryzacja według sumy bądź iloczynu mnogościowego).
Oczekuje się, że otrzymana macierz dopasowań będzie zawierać wyrazy, powiązane między sobą w ramach każdej pary zdań, będących wzajemnym tłumaczeniem.

Wewnętrzny algorytm symetryzacji odbywa się domyślnie po każdej iteracji każdego modelu statystycznego. W zależności od aktualnie uruchomionego modelu statystycznego (rysunek 16.), symetryzacja jest wykonywana jedynie na plikach prawdopodobieństwa tłumaczenia (t), bądź na plikach prawdopodobieństwa tłumaczenia (t), prawdopodobieństwa dopasowania pozycji (a) i prawdopodobieństwa zniekształcenia (d). Poszczególne schematy symetryzacji zostały przedstawione w algorytmie 1., algorytmie 2. i algorytmie 3.

Algorytm: Symetryzacja prawdopodobieństwa tłumaczenia (t)

Dla każdego dopasowania wyrazów o identyfikatorze i, j z pierwszego kierunku obliczeń, pobierz prawdopodobieństwo tłumaczenia.

Pobierz z przeciwnego kierunku obliczeń prawdopodobieństwo tłumaczenia wyrazów o identyfikatorze j, i.

Oblicz średnią arytmetyczną tych dwóch prawdopodobieństw.

Otrzymaną średnią wstaw do nowych tablic prawdopodobieństwa tłumaczenia wyrazów w obu kierunkach.

Wykonaj kolejną iterację danego modelu lub uruchom obliczenia dla nowego modelu.

Algorytm 1. Symetryzacja prawdopodobieństwa tłumaczenia (t).

Algorytm: Symetryzacja prawdopodobieństwa dopasowania pozycji (a)

Dla każdego powiązania wyrazów na pozycjach i, j z pierwszego kierunku obliczeń, pobierz prawdopodobieństwo dopasowania pozycji.

Pobierz z przeciwnego kierunku obliczeń prawdopodobieństwo dopasowania pozycji wyrazów na pozycjach j, i.

Oblicz średnią arytmetyczną tych dwóch prawdopodobieństw.

Otrzymaną średnią wstaw do nowych tablic prawdopodobieństwa dopasowania pozycji wyrazów w obu kierunkach.

Wykonaj kolejną iterację danego modelu lub uruchom obliczenia dla nowego modelu.

Algorytm 2. Symetryzacja prawdopodobieństwa dopasowania pozycji (a).

Algorytm: Symetryzacja prawdopodobieństwa zniekształcenia (d)

Dla każdego powiązania wyrazów na pozycjach j, i z pierwszego kierunku obliczeń, pobierz prawdopodobieństwo zniekształcenia.

Pobierz z przeciwnego kierunku obliczeń prawdopodobieństwo zniekształcenia wyrazów na pozycjach i, j.

Oblicz średnią arytmetyczną tych dwóch prawdopodobieństw.

Otrzymaną średnią wstaw do nowych tablic prawdopodobieństwa zniekształcenia wyrazów w obu kierunkach.

Wykonaj kolejną iterację danego modelu lub uruchom obliczenia dla nowego modelu.

Algorytm 3. Symetryzacja prawdopodobieństwa zniekształcenia (d).

3.1.3. Opis matematyczny dokonanych modyfikacji dla poszczególnych modeli

Symetryzacja dwukierunkowych modeli statystycznych dopasowania tekstu spowodowała zmiany w opisie matematycznym poszczególnych modeli. Poniższe podrozdziały szczegółowo opisują te zmiany i ich wpływ na dopasowania. Główne zmiany nastąpiły w końcowych równaniach prawdopodobieństw tłumaczenia i powiązania wyrazów. Wszystkie obliczenia dotyczące symetryzacji są wykonywane w głównym wątku algorytmu dopasowania wyrazów.

Model 1

Aby rozróżnić dwa równolegle obliczane modele dopasowań, zostaną one oznaczone jako α i β. Oznaczają on parametry dla pierwszego (od korpusu źródłowego do docelowego) i drugiego (od korpusu docelowego do źródłowego) kierunku obliczeń.

Dla modelu 1. otrzymujemy następujące równania prawdopodobieństwa warunkowego dwóch zdań **e** i **f**:

$$\Pr_{\alpha}(\mathbf{f} \mid \mathbf{e}) = \frac{\varepsilon(m \mid l)}{(l+1)^{m}} \sum_{a} \prod_{j=1}^{m} t_{\alpha}(f_j \mid e_{a_j}), \tag{40}$$

$$\Pr_{\beta}(\mathbf{e} \mid \mathbf{f}) = \frac{\varepsilon(l \mid m)}{(m+1)^{l}} \sum_{b} \prod_{i=1}^{l} t_{\beta}(e_i \mid f_{b_i}), \tag{41}$$

gdzie l i m są długościami zdania wejściowego i wyjściowego, a jest skierowanym dopasowaniem między zdaniami, t_{α} i t_{β} są skierowanymi prawdopodobieństwami tłumaczenia pomiędzy wyrazem źródłowym i docelowym (lub docelowym i źródłowym), a ε

stałą zależną od długości poszczególnych zdań, e i f oznaczają pojedyncze wyrazy, a i b oznaczają skierowane dopasowania pomiędzy zdaniami. Równania te interpretujemy jako prawdopodobieństwo, że maszyna tłumacząca dla zdania **e** zwróci zdanie **f** jako tłumaczenie i prawdopodobieństwo, że maszyna tłumacząca dla zdania **f** zwróci zdanie **e** jako tłumaczenie.

Prawdopodobieństwo tłumaczenia t_α i t_β przyjmuje następującą formę:

$$t_\alpha(f \mid e) = \frac{\sum_{S=1}^{S} c(f \mid e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}{\sum_{f'} \sum_{S=1}^{S} c(f' \mid e; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}, \tag{42}$$

gdzie S jest liczbą zdań w korpusie, $c(f \mid e; \mathbf{f}, \mathbf{e})$ jest częstością powiązań słowa wejściowego i wyjściowego w wyrazach ze zdania **f** i **e**.

W oryginalnym modelu częstość powiązań $c(f \mid e; \mathbf{f}, \mathbf{e})$ jest obliczana z wartości t za pomocą dwóch równań:

$$c(f \mid e; \mathbf{f}, \mathbf{e}) = \sum_a \Pr(a \mid \mathbf{f}, \mathbf{e}) \sum_{i,j} \delta(f, f_j) \delta(e, e_i), \tag{43}$$

$$\Pr(a \mid \mathbf{f}, \mathbf{e}) = \frac{\prod_{j=1}^{m} t(f_j \mid e_{a_j})}{\sum_a \prod_{j=1}^{m} t(f_j \mid e_{a_j})}, \tag{44}$$

gdzie δ jest deltą Kroneckera:

$$\delta(i, j) = \begin{cases} 1 & \text{jezeli i = j} \\ 0 & \text{w przeciwnym wypadku} \end{cases} \tag{45}$$

Modyfikacje zastosowane w modelu 1. powodują, że równanie (44) przyjmuje postać:

$$\Pr_\alpha(a \mid \mathbf{f}, \mathbf{e}) = \frac{\prod_{j=1}^{m} \tilde{t}(f_j \mid e_{a_j})}{\sum_a \prod_{j=1}^{m} \tilde{t}(f_j \mid e_{a_j})} = \frac{\prod_{j=1}^{m} (t_\alpha(f_j \mid e_{a_j}) + t_\beta(e_{a_j} \mid f_j))}{\sum_a \prod_{j=1}^{m} (t_\alpha(f_j \mid e_{a_j}) + t_\beta(e_{a_j} \mid f_j))}. \tag{46}$$

Suma po wszystkich prawdopodobieństwach t danego wyrazu e nadal sumuje się do 1:

$$\sum_f t(f \mid e) = 1. \tag{47}$$

To są jedyne różnice pomiędzy standardowym modelem 1. a modyfikacjami zastosowanymi w przypadku symetryzacji obliczeń dwukierunkowych. Dzięki wykorzystaniu prawdopodobieństwa tłumaczenia z obu kierunków obliczeń, informujemy każdy model o bardziej prawdopodobnym sposobie dopasowania. Jeśli słowo z języka źródłowego jest oczekiwanym tłumaczeniem słowa z języka docelowego, to to samo słowo z języka docelowego jest oczekiwanym tłumaczeniem słowa z języka źródłowego. Jeśli oba warunki są spełnione, to danemu tłumaczeniu możemy przypisać większe prawdopodobieństwo tłumaczenia.

Oczywiście częstości powiązań $c(f \mid e;\mathbf{f},\mathbf{e})$ i $c(e \mid f;\mathbf{e},\mathbf{f})$ przyjmują różne wartości dla tych samych słów i zdań. Jest to spowodowane różnym kierunkiem dopasowania wyrazów. W związku z tym w większości przypadków zachodzi nierówność:

$$t_\alpha(f \mid e) \neq t_\beta(e \mid f). \tag{48}$$

Model 2

W modelu 2. przy symetryzacji oprócz prawdopodobieństwa tłumaczenia, symetryzowane są także prawdopodobieństwa dopasowania pozycji a. Oznaczają one prawdopodobieństwo, że dla danej długości obu zdań, wyraz docelowy na pozycji j jest powiązany z wyrazem źródłowym na pozycji a_j.

Prawdopodobieństwo skierowane dla tego modelu jest równe:

$$\Pr_\alpha(\mathbf{f} \mid \mathbf{e}) = \varepsilon(m \mid l)\sum_a \prod_{j=1}^{m} (t_\alpha(f_j \mid e_{a_j}) a_\alpha(a_j \mid j,m,l)). \tag{49}$$

Równania (42) i (43) dla modelu 2. nie ulegają zmianie, natomiast nowy parametr - prawdopodobieństwo dopasowania pozycji przyjmuje postać:

$$a_\alpha(i \mid j,m,l) = \frac{\sum_{S=1}^{S} c(i \mid j,m,l;\mathbf{f}^{(s)},\mathbf{e}^{(s)})}{\sum_{i'}\sum_{S=1}^{S} c(i' \mid j,m,l;\mathbf{f}^{(s)},\mathbf{e}^{(s)})}, \tag{50}$$

gdzie

$$c(i \mid j,m,l;\mathbf{f},\mathbf{e}) = \sum_a \Pr_\alpha(a \mid \mathbf{f},\mathbf{e})\delta(i,a_j). \tag{51}$$

Prawdopodobieństwo $\Pr_\alpha(a \mid \mathbf{f}, \mathbf{e})$ przyjmuje postać:

$$\Pr_\alpha(a \mid \mathbf{f}, \mathbf{e}) = \frac{\prod_{j=1}^{m} (\tilde{t}(f_j \mid e_{a_j}) \tilde{a}(a_j, j, m, l))}{\sum_a \prod_{j=1}^{m} (\tilde{t}(f_j \mid e_{a_j}) \tilde{a}(a_j, j, m, l))}, \tag{52}$$

gdzie $\tilde{t}$ jest prawdopodobieństwem tłumaczenia obliczonym z sumy prawdopodobieństw tłumaczenia w obu kierunkach obliczeń, podobnie jak w modelu 1.; $\tilde{a}$ jest prawdopodobieństwem dopasowania pozycji wyliczonym w następujący sposób:

$$\tilde{a}(i \mid j, l, m) = a_\alpha(i \mid j, m, l) + a_\beta(j \mid i, l, m) \ . \tag{53}$$

Suma po długości zdania wejściowego prawdopodobieństw a sumuje się do 1:

$$\sum_{i=0}^{l} a(i \mid j, m, l) = 1. \tag{54}$$

Dodatkowo, dzięki symetryzacji zarówno prawdopodobieństw dopasowania pozycji, jak i prawdopodobieństw tłumaczenia, efekt wymiany informacji pomiędzy dwoma modelami wzrasta.

Model HMM

Jak wspominaliśmy, model HMM jest stosowany zamiast mniej efektywnego modelu 2. Formuła prawdopodobieństwa warunkowego dla tego modelu jest bardzo zbliżona do modelu 2.:

$$\Pr(\mathbf{f} \mid \mathbf{e}) = \varepsilon(m \mid l) \sum_a \prod_{j=1}^{m} (t(f_j \mid e_{a_j}) a(a_{j-1}, l)), \tag{55}$$

która jest bardzo zbliżona do równania (10). Prawdopodobieństwo dopasowania pozycji a_j na pozycji j zależy od poprzedniego prawdopodobieństwa pozycji a_{j-1}. Dzięki takiemu zabiegowi model dopasowania został przekształcony w model Markowa pierwszego rzędu. Prawdopodobieństwo pozycji w takim przypadku dla pierwszego kierunku obliczeń jest zdefiniowane następująco:

$$a_\alpha(i \mid i', l) = \frac{\sum_{S=1}^{S} c(i \mid i', l; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}{\sum_{i''} \sum_{S=1}^{S} c(i'' \mid i', l; \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}, \tag{56}$$

$$c(i \mid i', l; \mathbf{f}, \mathbf{e}) = \sum_a \Pr_\alpha(a \mid \mathbf{f}, \mathbf{e}) \sum_j (\delta(i', a_{j-1}) \delta(i, a_j)). \tag{57}$$

Prawdopodobieństwo tłumaczenia t i powiązane miary pozostają takie same jak w modelu 1. i 2. Natomiast definicja równania:

$$\Pr_{\alpha}(a \mid \mathbf{f}, \mathbf{e}) = \frac{\prod_{j=1}^{m}(t_{\alpha}(f_j \mid e_{a_j})a_{\alpha}(a_j \mid a_{j-1}, l))}{\sum_{a}\prod_{j=1}^{m}((t_{\alpha}(f_j \mid e_{a_j})a_{\alpha}(a_j \mid a_{j-1}, l))} \tag{58}$$

jest zastąpiona przez wzór:

$$\Pr_{\alpha}(a \mid \mathbf{f}, \mathbf{e}) = \frac{\prod_{j=1}^{m}(\tilde{t}(f_j \mid e_{a_j})a(a_j \mid a_{j-1}, l))}{\sum_{a}\prod_{j=1}^{m}((\tilde{t}(f_j \mid e_{a_j})a(a_j \mid a_{j-1}, l))}. \tag{59}$$

Prawdopodobieństwo dopasowania pozycji a dla tego modelu pozostaje bez zmian. Dla modelu 2. prawdopodobieństwo to można było zmienić poprzez podmianę wartości źródłowych z docelowymi. Taka zmiana dla modelu Markowa zmieniałaby interpretację prawdopodobieństwa dopasowania. W tym przypadku lepszym rozwiązaniem będzie dopasowanie jedynie wyrazów w jednym kierunku. Natomiast dla modelu HMM nadal pozostaje symetryzacja prawdopodobieństw tłumaczenia.

Model 3., 4. i 5.

Modyfikacje dla modelu 3., 4. i 5. bazują na tych samych spostrzeżeniach, co w modelach opisanych powyżej. Kolejne modele bazują na zmianach prawdopodobieństw tłumaczenia, dopasowania pozycji i zniekształcenia zgodnie z rysunkiem 16.

3.2. Zastosowanie symetryzacji na końcu obliczeń

W końcowym etapie obliczeń powstają dwie macierze wynikowe dopasowań wyrazów z dwóch kierunków. Obie zwierają jedynie powiązania jeden do wielu (jeden wyraz ze zdania wejściowego może mieć 1 lub więcej powiązań z wyrazami ze zdania docelowego). Nie ma natomiast powiązań wiele do jednego, a tym bardziej wiele do wielu. Aby otrzymać powiązania wiele do wielu, należy wykonać symetryzację na końcu procesu obliczeń. Jedną z najpopularniejszych i efektywnych metod symetryzacji jest metoda *refined* opisana w rozdziale 2.2.2.

3.3. Zastosowanie metody przyśpieszenia obliczeń – wielowątkowość, zoptymalizowane algorytmy symetryzacji

Dzięki zastosowaniu wielowątkowości oba kierunki dopasowywania wyrazów mogą być przetwarzane jednocześnie. To znacznie skraca czas obliczeń, zwłaszcza dla maszyn wielordzeniowych.

Ponadto zoptymalizowano algorytmy symetryzacji prawdopodobieństw tłumaczenia, dopasowania i zniekształcenia dla dwóch kierunków dopasowania. Symetryzacja następuje na wynikowych tablicach prawdopodobieństw, które po symetryzacji są tablicami wejściowymi kolejnej iteracji danego modelu lub iteracji kolejnego modelu.

3.4. SyMGIZA++ - narzędzie dopasowania tekstów z wykorzystaniem wielowątkowości i symetryzacji obliczeń

Omówione powyżej rozwiązania dotyczące symetryzacji dwukierunkowych dopasowań wyrazów metodami statystycznymi zostały zaimplementowane w narzędziu SyMGiza++. Narzędzie to jest rozwinięciem narzędzia dopasowania wyrazów – MGiza++. Wykorzystuje zalety MGizy++, czyli wielowątkowość i dodatkowo implementuje nowe mechanizmy symetryzacji dwukierunkowych dopasowań, zaprezentowanych w tym rozdziale. Algorytm działania narzędzia przedstawia rysunek 16.

SyMGiza++ została napisana w języku C/C++. Dodatkowo powstały skrypty bash ułatwiający pracę z narzędziem. Całość jest publicznie dostępna w postaci kodów źródłowych z możliwością kompilacji pod każdym systemem operacyjnym. Aplikacja wraz z opisem instalacji i konfiguracji znajduje się pod adresem:
http://psi.amu.edu.pl/en/index.php?title=Downloads.

3.5. Złożoność obliczeniowa algorytmów zawartych w narzędziu SyMGIZA++

SyMGiza++ jest wzorowana na narzędziu Giza++ i rozszerzeniu MGiza++. Z tego względu złożoność obliczeniowa zastosowanych algorytmów jest zbliżona do złożoności algorytmów zawartych w tych narzędziach. Główna różnica polega na tym, że SyMGiza++ wykonuje obliczenia w dwóch kierunkach i z tego względu zarówno czas obliczeń, jak i niezbędna pamięć ulega podwojeniu.

Złożoność obliczeniowa dla modelu 1. i 2. wynosi $O(l \cdot m)$, dla modelu HMM $O(l + m^2)$ [19], gdzie l oznacza ilość wyrazów w zdaniu źródłowym natomiast m - ilość wyrazów

w zdaniu docelowym. Do tego dochodzi jeszcze złożoność poszczególnych kroków algorytmu EM opisanego w paragrafie 1.4. Złożoność obliczeniowa kroku E rośnie liniowo z ilością par zdań korpusu dwujęzycznego: $O(S)$. W kroku M złożoność obliczeniowa jest zdominowana przez normalizację prawdopodobieństw słownikowych. W najgorszym przypadku wynosi ona $O(|V_F| \cdot |V_E|)$ [19], gdzie $|V_F|$ jest wielkością słownika źródłowego, a $|V_E|$ - wielkością słownika wyjściowego. Czas obliczeń kroku M jest więc wielomianowy w stosunku do długości słownika, który rośnie z kolei logarytmicznie do wielkości korpusu $O(\log S)$. Wynika stąd, że złożoność obliczeniowa kroku M wynosi $O(S)$, a dla całego algorytmu EM wynosi również $O(S)$.

Złożoności obliczeniowe poszczególnych kroków obliczeń narzędzi Giza++ i MGiza++ wynoszą $O(S_l + S_m^2)$, gdzie S_l oznacza liczbę wszystkich wyrazów w korpusie źródłowym, a S_m liczbę wyrazów w korpusie docelowym. Tym sposobem otrzymujemy złożoność kwadratową do ilości wyrazów w korpusie docelowym. Taką samą złożoność otrzymujemy dla algorytmów zawartych w narzędziu SyMGiza++.

Rozdział 4

Ewaluacja wyników

4.1. Środowisko testowe

Zanim zostały rozpoczęte prace nad rozwojem narzędzi dopasowania wyrazów, przeanalizowano wydajność oraz jakość istniejących rozwiązań. Wykonano testy szybkości obliczeń i jakości dopasowań wyrazów z użyciem trzech narzędzi: Giza++, MGiza++ i PGiza++. Prezentowane wyniki zostały wygłoszone na konferencji w Gnieźnie w 2010 roku [24]. Okazało się, że jakość dopasowań dla tych wszystkich narzędzi jest jednakowa. Jednak czas obliczeń znacząco się różnił.

W obliczeniach wykorzystano klaster IBM Blade, złożony z siedmiu węzłów. Tabela 10. przedstawia pełną specyfikację klastra. Do klastra podłączono macierz dyskową z podłączonym zasobem o pojemności 50GB.

Wyposażenie:	Węzły 1, 2, 3, 4, 5 (konfiguracja I)	Węzły 6, 7 (konfiguracja II)
• **processor**	2 x Intel(R) Xeon(TM) CPU 3.06GHz (każdy ma 2 rdzenie)	2 x Intel(R) Xeon(R) 2,5GHz (każdy ma 4 rdzenie)
• **pamięć**	2,5GB	8GB
• **dysk twardy**	Macierz dyskowa współdzielona o pojemności 50GB	Macierz dyskowa współdzielona o pojemności 50GB
• **karta sieciowa**	Macierz FC o prędkości 2GBit, połączona interfejsami 1GBit	Macierz FC o prędkości 2GBit, połączona interfejsami 1GBit
• **system operacyjny**	Debian GNU 5.0, 32-bitowa maszyna	Debian GNU 5.0, 64-bitowa maszyna

Tabela 10. Konfiguracja sprzętowa klastra obliczeniowego.

Eksperymenty przeprowadzono najpierw na niewielkich korpusach tekstowych, co pozwoliło wykonać szybkie testy poprawności kompilacji poszczególnych programów. Następnie wykonano obliczenia dla dużych korpusów, aby odkryć potencjał drzemiący w poszczególnych rozwiązaniach.

Wszystkie użyte w porównaniach aplikacje zostały skompilowane na klastrze z użyciem kompilatora języka C++: G++ w wersji 4.2.4. Kompilacje zostały wykonywane równolegle dla węzłów 1-5 i węzłów 6-7. Czasy obliczeń Gizy++ i MGizy++ przedstawiono dla konfiguracji I i II. PGiza++ została uruchomiona na obu częściach klastra (węzły 1-5, węzły 6-7).

4.2. Szybkość działania poszczególnych narzędzi

4.2.1. Porównanie narzędzi: Giza++, MGiza++, PGiza++

Przy wyznaczaniu czasu obliczeń brano pod uwagę całkowity czas, potrzebny do uzyskania z korpusów dwujęzycznych gotowego słownika. Wymagało to utworzenia odpowiednich plików, wykorzystywanych w dalszych etapach obliczeń. We wszystkich obliczeniach wykonano 5 iteracji modelu 1., 5 iteracji modelu HMM, 5 iteracji modelu 3. i 5 iteracji modelu 4. [9], [11]. Tabela 11. przedstawia czasy obliczeń plików wejściowych dla korpusu polsko-francuskiego, wielkości około 100MB każdy plik (tłumaczenie w kierunku polsko-francuskim). Opis typów obliczeń wraz z komendami wykonującymi te obliczenia został przedstawiony w dodatku B.

Typ obliczeń:	Giza++ (węzeł 1)	Giza++ (węzeł 6)	MGiza++ (węzeł 1)	MGiza++ (węzeł 6)
• plain2snt	24m 38.396s	3m 8.161s	26m 51.139s	2m 55.109s
• mkcls	66m 54.497s + 26m 20.822s	23m 36.562s + 10m 56.799s	63m 16.045s + 25m 53.571s	23m 40.987s + 10m 49.471s
• snt2cooc	19m 48.488s	7m 22.989s	14m 24.183s	5m 15.085s
Czas całkowity	137m 42.203s	45m 4.511s	130m 24.938s	42m 40.652s

Tabela 11. Czas przygotowania plików wejściowych dla korpusu zawierającego 748 734 zdań w języku polsko-francuskim.

Wszystkie obliczenia były wykonywane na nieobciążonym klastrze, aby wyeliminować ewentualne opóźnienia spowodowane kolejkowaniem dostępu do dysku lub czasu procesora.

Tabela 12. przedstawia wyniki obliczeń dla małych korpusów danych tłumaczenia angielsko-niemieckiego (dwa korpusy danych wielkości około 700KB). Liczba zdań do porównania to 6088. Czasy przedstawione w tabeli (Giza++, MGiza++) dotyczą jedynie czasu

potrzebnego na wykonanie odpowiednich modeli statystycznych. Czasy przygotowania odpowiednich plików wejściowych nie zostały uwzględnione, gdyż nie zmieniają one znacząco czasów obliczeń. Natomiast dla PGiza++ wzięto pod uwagę całkowite czasy obliczeń, zawierające również czasy utworzenia odpowiednich plików wejściowych.

Giza++ (węzeł 1)	Giza++ (węzeł 6)	MGiza++ (węzeł 1)	MGiza++ (węzeł 6)	PGiza++ (węzeł 6 i 7)
757s	160s	4 wątki – 264s 8 wątków – 254s	1 wątek – 182s 4 wątki – 80s 8 wątków – 83s	2 procesy – 25m 29s 6 procesów -36m 19s

Tabela 12. Czas obliczeń dla korpusu zawierającego 6088 zdań w języku angielsko-niemieckim.

Jak widzimy, istnieje ogromna różnica między czasami obliczeń narzędzia Giza++, MGiza++ a PGiza++ dla tego korpusu. Na niekorzyść narzędzia PGiza++ wpływa kilku czynników:

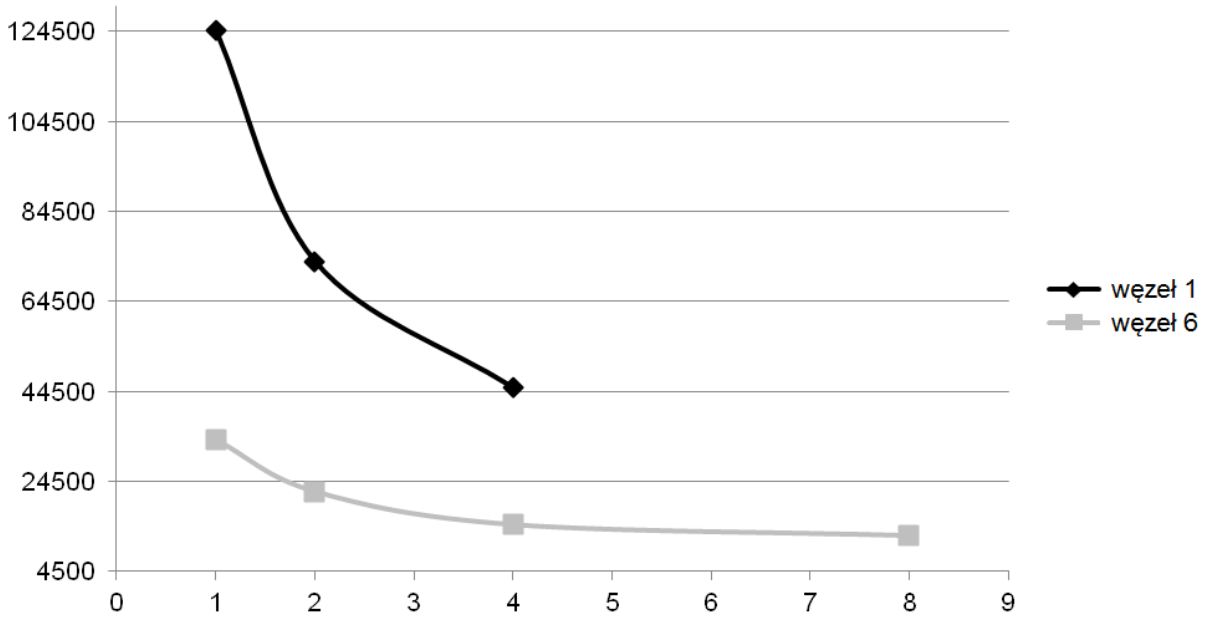
- korpusy danych są bardzo małe, przez to w PGiza++ dużo czasu zajmuje czas komunikacji i synchronizacji poszczególnych procesów,
- dodatkowo, wymagana jest normalizacja wyników po każdej pętli iteracji,
- PGiza++ tworzy automatycznie pliki wejściowe, które nie zostały uwzględnione w czasach obliczeń Giza++ i MGiza++.

Kolejnym etapem testów było wykonanie obliczeń na dużych korpusach danych. Zastosowano korpus polsko-francuski o wielkości około 100MB dla każdego kierunku, zawierający 748 734 zdań. Tabela 11. zawiera czasy obliczeń początkowych dla tego korpusu. Czasy te należy dodać do całkowitych czasów potrzebnych do uzyskania końcowej macierzy dopasowania. Tabela 13. przedstawia sumaryczne wyniki. Obliczenia wykonano osobno dla węzłów 1-5 i 6-7. Wynika to z innej specyfikacji sprzętowej tych węzłów, a co za tym idzie różnych czasów obliczeń. Ponadto uwzględniono tu również czas obliczeń dla narzędzi SyMGiza++.

Nazwa programu	Ilość wątków	Czas obliczeń początkowych	Czas obliczeń programu głównego	Sumaryczny czas obliczeń
Giza++ (węzeł 1)	1	137m 42s	1770m 23s	1908m 5s
Giza++ (węzeł 6)	1	45m 4s	483m 56s	529m
MGiza++ (węzeł 1)	4	130m 24s	625m 6s	755m 30s
MGiza++ (węzeł 1)	2	130m 24s	1092m 23s	1222m 47s
MGiza++ (węzeł 1)	1	130m 24s	1947m 33s	2077m 57s
MGiza++ (węzeł 6)	8	42m40s	164m 34s	207m 14s
MGiza++ (węzeł 6)	4	42m40s	205m 20s	248m
MGiza++ (węzeł 6)	2	42m40s	327m 15s	369m 55s
MGiza++ (węzeł 6)	1	42m40s	517m 58s	560m 38s
PGiza++ (węzły 1 - 5)	5	-	-	1388m 5s
PGiza++ (węzły 6 i 7)	2	-	-	850m 15s
PGiza++ (węzły 3*6 i 3*7)	6	-	-	762m 18s
SyMGiza++ (węzeł 1)	4	-	-	1592m12s
SyMGiza++ (węzeł 6)	8	-	-	421m18s

Tabela 13. Czas obliczeń dla korpusu zawierającego 748734 zdań w języku polsko-francuskim.

Jak można, zauważyć czasy obliczeń Giza++ i MGiza++ na jednym procesorze są podobne. Jednak zwiększenie liczby wątków do 4 pozwala zredukować czas obliczeń o ponad połowę. Rysunek 17. przedstawia przyśpieszenie obliczeń w zależności do ilości wątków.



Rysunek 17. MGIZA++ - zależność szybkości obliczeń (w sekundach) od ilości wątków

Porównując otrzymane wyniki z pracą Q. Gao i S. Vogel [19] otrzymujemy podobne średnie czasy przyśpieszenia obliczeń dla:

- 2 procesorów - w granicach 59%-65% czasu dla jednego procesora,
- 4 procesorów – 36% - 41% czasu dla jednego procesora

w zależności od środowiska testowego.

Jeśli chodzi o obliczenia wykonane za pomocą PGiza++, to wyniki nie są już tak dobre. Przyśpieszenie obliczeń na węźle 6 i 7 jest nieznaczne w porównaniu z Giza++. Natomiast zwiększenie ilości wątków na poszczególnych węzłach nie powoduje polepszenia czasów obliczeń.

Sytuacja lepiej wygląda na węzłach 1-5. Tam obserwujemy przyśpieszenie obliczeń, lecz jest ono nieznaczne, biorąc pod uwagę użycie pięciu procesorów. Jest to związane głównie z nierównomiernym zakończeniem procesów na poszczególnych węzłach. Główny proces musi czekać na zakończenie wszystkich podprocesów i dopiero wtedy może wykonać normalizację, po czym uruchomić kolejną iterację. Normalizacja jest wykonywana na jednym z węzłów i zajmuje też sporą część obliczeń. Porównując otrzymane wyniki z pracą Q. Gao i S. Vogel [19] dochodzimy do wniosku, że otrzymane w moich obliczeniach przyśpieszenie

jest znacznie mniejsze (1.37-krotnie) od przyspieszenia z powyższej pracy (4.3-krotne). Wynika to z kilku aspektów. W zacytowanej pracy stosowano jeszcze większe korpusy danych, na których obliczenia trwały kilka dni, a obliczenia były wykonywane na innej specyfikacji sprzętowej. Dodatkowo zastosowano mniejszą ilość iteracji modelu 3. i 4. Nasuwa się więc wniosek, że im większy korpus, tym PGiza++ szybciej wykonuje obliczenia w porównaniu z Giza++.

Jeśli chodzi o jakość wyników, to wszystkie trzy wersje narzędzia (Giza++, MGiza++ i PGiza++) produkują identyczne powiązania. PGiza++ najlepiej działa na dużych korpusach, gdzie jest potrzebna duża ilość obliczeń, a czas normalizacji jest znacząco mniejszy od czasu obliczeń na poszczególnych węzłach. Dodatkowo PGiza++ produkuje w czasie działania dużo plików tymczasowych, które powodują drastyczny wzrost potrzeby na wolną przestrzeń dyskową. Sposób działania tego algorytmu powoduje duże straty czasu na normalizację wyników po każdej iteracji, a także oczekiwanie na zakończenie wszystkich obliczeń danego kroku. Te wady powodują, że PGiza++ nie jest wydajnym rozwiązaniem. Natomiast bardzo przydatne okazują się jej mechanizmy komunikacji pomiędzy węzłami oraz współdzielenia plików.

Z powyższych wyników można wywnioskować, że MGiza++ jest najlepszym rozwiązaniem. Dobrze sprawuje się zarówno na małych, jak i dużych korpusach. Zużywa podobną ilość zasobów pamięci RAM i dyskowej, co Giza++, a ponadto w pełni wykorzystuje możliwości procesorów wielordzeniowych. Z moich obliczeń wynika, że najlepszą metodą zwiększenia prędkości obliczeń jest wykorzystanie maszyny, która posiada:

- jak największą ilość rdzeni/procesorów,
- jak najbardziej wydajne rdzenie.

Powyższe wyniki utwierdzają w przekonaniu, że narzędzia wykorzystujące wielordzeniowość współczesnych komputerów pozwalają znacznie zmniejszyć czas obliczeń. MGiza++ wydaje się odpowiednim programem do tworzenia słowników z dwujęzycznych korpusów. Potwierdziły się również przypuszczenia, że PGiza++ nie sprawdza się w optymalizacji szybkości. Stosowane współcześnie coraz większe korpusy danych, wymagają utworzenia narzędzia, umożliwiającego w miarę szybkim czasie uzyskać wyniki.

Ostatnim etapem jest porównanie wyników otrzymanych w narzędziu SyMGiza++. Jeśli chodzi o jakość wyników, to przedstawione są one szczegółowo w rozdziale 4.3.1. Natomiast szybkość działania narzędzia pokazuje, że jest ono zbliżone do szybkości MGiza++. W tabeli 13. widać, że czas obliczeń jest ponad dwukrotnie dłuższy dla SyMGiza++

w porównaniu z MGiza++, jednak w tym przypadku wykonywane jest dwa razy więcej obliczeń – symetryzacja obliczeń z dwóch kierunków dopasowywania.

Podsumowując powyższe eksperymenty, dochodzimy do wniosku, że korzystając z mechanizmów PGiza++ i możliwości MGiza++ można stworzyć wydajne narzędzie dopasowywania wyrazów w dużych korpusach danych.

Porównania w narzędziu MGiza++ i autorskim rozwiązaniu SyMGiza++ pokazały, że jeśli chodzi o czas obliczeń, są one bardzo zbliżone. Dłuższy czas obliczeń dla narzędzi SyMGiza++ wynika z nakładu czasu potrzebnego na wykonanie symetryzacji po każdym kroku iteracji.

4.3. Porównanie wyników z dostępnymi publikacjami – współczynnik AER, BLEU

Aby sprawdzić jakość otrzymanych wyników dopasowania wyrazów z użyciem symetryzacji posłużono się dwoma miarami odzwierciedlającymi trafność dopasowań (AER) i jakość tłumaczenia (BLEU). Szczegóły dotyczące każdej metody zostały przedstawione w rozdziałach 2.3 i 2.4.

4.3.1. Wyniki otrzymane z wykorzystaniem narzędzia SyMGiza++ i symetryzacji końcowej

W przypadku miary trafności dopasowania AER posłużono się korpusem francusko-angielskim, utworzonym podczas konferencji HTL-NAACL 2003 [41]. Testy zostały wykonane na czterech narzędziach Giza++, MGiza++, SyMGiza++ i BerkleyAligner [42], [43]. Dla każdego z nich wykonano obliczenia w obu kierunkach: od korpusu francuskiego do angielskiego i od korpusu angielskiego do francuskiego. Materiał prezentujący wyniki został przedstawiony podczas konferencji w Wiśle w 2010 roku [44]. Schemat obliczeń składał się z:

- 5 iteracji modelu 1.,
- 5 iteracji modelu Markowa – HMM,
- 5 iteracji modelu 3. i
- 5 iteracji modelu 4.

Dla narzędzia BerkleyAligner zostały użyte standardowe ustawienia ze współdzielonym modelem.

Po zakończeniu obliczeń dwukierunkowe macierze dopasowań (przykład 10.) połączono za pomocą symetryzacji *refined* (rozdział 2.2.2).

Wszystkie obliczenia wykonano na maszynie zawierającej 4 rdzenie procesora i 8 GB pamięci RAM.

Użyte narzędzie	Czas obliczeń [min]	Precyzja (definicja 17.) [%]	Pokrycie (definicja 18.) [%]	AER [%]
Giza++ EN-FR	-	91,19	92,20	8,39
Giza++ FR-EN	-	91,82	87,96	9,79
Giza++ REFINED	457	93,24	92,59	7,02
MGiza++ EN-FR	-	91,19	92,22	8,40
MGiza++ FR-EN	-	91,84	87,96	9,78
MGiza++ REFINED	306	93,25	92,60	7,01
BerkleyAligner	-	90,74	95,99	7,24
SyMGiza++	332	94,34	94,08	**5,76**

Tabela 14. Wynik obliczeń współczynnika AER na korpusie francusko-angielskim powstałym podczas konferencji HTL-NAACL 2003.

Jak widać z powyższej tabeli, SyMGiza++ uzyskała lepszy (niższy) współczynnik AER niż wszystkie dotychczasowe narzędzia.

Czas obliczeń dla poszczególnych narzędzi jest liczony od uruchomienia programu aż do zakończenia procesu symetryzacji. Wyniki obliczeń przedstawiono w tabeli 14. Nie powinno dziwić, że współczynnik AER jest taki sam dla Gizy++ jak i dla MGizy++. MGiza++ jest o około 33% szybsza niż Giza++. Jest również szybsza niż SyMGiza++. SyMGiza++ daje najlepszy współczynnik AER. Polepsza trafność dopasowania o ponad 17% w porównaniu z Gizą++ i MGizą++. Poprawia także współczynnik AER o ponad 20% w porównaniu z BerkleyAligner. Podczas przeprowadzonych eksperymentów nie udało się powtórzyć wyników BerkleyAligner raportowanych w pracy [42], gdzie współczynnik AER wynosi 4,9%. Natomiast wyniki prezentowane w pracy [42] dotyczące Giza++ są praktycznie identyczne z powyższymi.

Aby sprawdzić jakość tłumaczenia automatycznego, użyto miary BLEU. Wykorzystano narzędzie Moses (1.5.2) skonfigurowane zgodnie z wytycznymi ACL 2008 Third Workshop on Statistical Machine Translation (WMT-08) – Shared Translation Task. Wyniki i szczegóły

dotyczące środowiska testowego przedstawiono w artykule [45]. Podczas obliczeń użyto różnych par języków i kierunków obliczeń. Modyfikacja systemu Moses dla SyMGizy++ polegała jedynie na zastąpieniu domyślnej Gizy++ tym narzędziem. Wyniki obliczeń miary BLEU dla różnych korpusów zaprezentowano w tabeli 15., tabeli 16. i tabeli 17.

	FR-EN	EN-FR	ES-EN	EN-ES	DE-EN	EN-DE	ES-DE	DE-ES
MGiza++	0,3189	0,2944	0,3241	0,03184	0,2656	0,1982	0,1996	0,2706
SyMGiza++	0,3193	**0,3000**	0,3231	0,3172	0,2657	0,1993	0,2014	**0,2741**

Tabela 15. Miara BLEU dla korpusu z oficjalnymi tekstami europejskimi – Europarl (test2008)

	FR-EN	EN-FR	ES-EN	EN-ES	DE-EN	EN-DE	ES-DE	DE-ES	CZ-EN	EN-CZ
MGiza++	0,2565	0,2339	0,3367	0,3220	0,2305	0,1531	0,1233	0,1775	0,2281	0,1285
SyMGiza++	0,2630	0,2359	0,3380	0,3234	0,2381	**0,1575**	0,1234	**0,1822**	**0,2369**	**0,1329**

Tabela 16. Miara BLEU dla korpusu z krótkimi komentarzami – News Commentary (nc-test2008)

	HU-EN	EN-HU
MGiza++	0,0587	0,0449
SyMGiza++	**0,0632**	0,0458

Tabela 17. Miara BLEU dla korpusu z potocznym językiem angielskim – Hunglish (newstest2008)

Pogrubioną czcionką zaznaczono miary, które uległy polepszeniu w sposób istotny ze statystycznego punktu widzenia. Pojęcie istotności ze statystycznego punktu widzenia zostało rozwinięte w pracy [46].

Miara tłumaczenia BLEU dla SyMGizy++ w większości przypadków jest lepsza niż dla standardowej MGizy++. Wyjątek stanowi jedynie tłumaczenie dwukierunkowe es-en i en-es. Są to jednak różnice nieistotne ze statystycznego punktu widzenia.

Polepszenie jakości tłumaczenia jest głównie widoczne dla małych korpusów (New commentary), które zawierają około 70 000 par zdań. Korpus równoległy Europarl ma ponad milion par zdań, podobnie jak korpus Hunglish.

Rozdział 5

Wnioski i uwagi

5.1. Podsumowanie otrzymanych wyników

Powyższa praca przedstawia nowe spojrzenie na proces dopasowania wyrazów metodami statystycznymi z użyciem symetryzacji obliczeń. W dotychczasowych pracach symetryzację stosowano jedynie po procesie dopasowania wyrazów i wszystkie obliczenia wykonywano na plikach wynikowych procesu dopasowania. W tej pracy zaprezentowano podejście, w którym symetryzacja następuje wewnątrz procesu dopasowania wyrazów. Po każdej iteracji kolejnych modeli statystycznych można wykonać symetryzację części danych, aby kolejna iteracja zawierała większą ilość informacji o najlepszych dopasowaniach wyrazów. Jak pokazały badania, takie podejście do problemu zwiększa trafność dopasowania wyrazów (zmierzoną za pomocą współczynnika AER). Poprawie uległa również jakość tłumaczenia obliczona za pomocą miary BLEU.

Zaimplementowane mechanizmy symetryzacji wymagają jeszcze dopracowania pod względem efektywności działania. Zauważono, że w zależności od częstości wykonywania symetryzacji (np. co piątą iterację danego modelu) i współczynnika wagi danej symetryzacji, otrzymane wyniki znacznie się różnią. Ta zależność wymaga wnikliwego sprawdzenia i opisania zarówno od strony matematycznej, jak i obliczeniowej.

5.2. Plan dalszych prac

Jednym z kierunków dalszych prac będzie próba wyważenia zależności najlepszego dopasowania i najlepszej jakości tłumaczenia od częstości symetryzacji i współczynnika wagi symetryzacji.

Kolejnym problemem wymagającym dalszego rozwoju jest szybkość obliczeń. Aby przyśpieszyć obliczenia, trzeba po raz kolejny sięgnąć po rozwiązania zaproponowane w PGiza++ [19], [20], a rozwinięte w narzędziu Chaski [21], czyli wieloprocesorowość obliczeń z wykorzystaniem wielu maszyn kilku rdzeniowych. Zaimplementowanie symetryzacji w takim narzędziu może spowodować poprawę szybkości, bez wpływu na jakość dopasowania wyrazów.

Trzecią drogą rozwoju danego problemu jest rozwinięcie symetryzacji wewnątrz procesu obliczeń, poprzez zastosowanie odmiennych algorytmów łączenia prawdopodobieństw z dwóch kierunków dopasowywania wyrazów. W tej chwili wykorzystuje się prosty mechanizm iloczynu mnogościowego (rozdział 2.2.2), który można by zastąpić bardziej wydajnym algorytmem *refined*.

Ostatnim pomysłem, który wydaje się ciekawy do dalszego badania, jest możliwość utworzenia nowego modelu statystycznego, który wykorzystywałby zalety dotychczasowych modeli, a dodatkowo obliczenia byłyby w nim wykonywane dwukierunkowo.

5.2.1. Optymalizacja algorytmów symetryzacji

Przedstawione w pracy algorytmy zostały zoptymalizowane pod względem szybkości działania, jednak można jeszcze zoptymalizować je pod względem przekazywanej informacji. Poprzez zastosowanie bardziej wyrafinowanych metod symetryzacji dla poszczególnych prawdopodobieństw tłumaczenia, dopasowania pozycji i zniekształcenia można kolejnym iteracjom danego modelu przekazać większą ilość informacji o dopasowaniach wykonanych w przeciwnym kierunku. Odpowiednie wyważenie tych informacji powinno zwiększyć trafność dopasowań, co może również poprawić jakość tłumaczenia.

5.2.2. Wykonanie obliczeń na dużych korpusach danych

Przeprowadzone do tej pory obliczenia były wykonywane na stosunkowo dużych korpusach zawierających około 750 tysięcy zdań, jednak korpusy takie mogą również liczyć po kilka milionów zdań. Ze względu na ograniczone zasoby sprzętowe i czasowe nie udało się wykonać obliczeń dla tak dużych korpusów. Jednak przygotowanie maszyny z dużą ilością rdzeni i odpowiednią ilością pamięci RAM może znacznie przyśpieszyć czas obliczeń. Te obliczenia ukazałyby, jak poprawia się trafność dopasowań i jakość tłumaczenia, gdy mamy do czynienia z korpusami dużej wielkości.

5.2.3. Zastosowanie innych technik przyśpieszenia obliczeń

Odpowiednie zarządzanie komunikacją pomiędzy węzłami, a także odpowiednio zmodyfikowane algorytmy modeli statystycznych pozwolą znacznie przyśpieszyć obliczenia.

Dodanie symetryzacji do obliczeń równoległych nie zwiększa drastycznie czasu obliczeń, a otrzymane wyniki są takie same jak dla obliczeń szeregowych.

Bibliografia

[1] European Parliament Proceedings Parallel Corpus 1996-2006: http://www.statmt.org/europarl/

[2] W. Weaver, "Transaltion", Machine Translation of Languages, MIT Press, 1955

[3] P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Mercer, P. Roossin, "A statistical approach to language translation", Proceedings of 12th International Conference on Computational Linguistics (COLING-88), Budapest, Hungary, 1988

[4] P. Brown, J. Cooke, S. Della Pietra, V. Della Pietra, F. Jelinek, J. Lafferty, R. Mercer, P. Roossin, "A statistical approach to machine translation", Computational Linguistics, 1990

[5] S. Warwick, G. Russell, "Bilingual concordancing and bilingual lexicography", Proceedings of EURALEX 4th International Congress, Malaga, Spain, 1990

[6] P. Brown, J. Lai, R. Mercer, "Aligning sentences in parallel corpora", Proceedings of 29th Annual Meeting of the Association for Computational Linguistics, Berkley CA, 1991

[7] W. Gale, K. Church, "A program for aligning sentences in bilingual corpora", Proceedings of 29th Annual Meeting of the Association for Computational Linguistics, Berkley CA, 1991

[8] M. Kay, "Text-translation alignment", ACH/ALCC '91: "Making Connections" Conference Handbook, 1991

[9] P.F. Brown, V.J.D. Pietra, S.A.D Pietra, R.L Mercer: The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics 19(2) 1993

[10] Al-Onaizan, Y., Curin, J., Jahr, M., Knight, K., Larerty, J., Melamed, I., Och, F., Purdy, D., Smith, N., Yarowsky, D.: Statistical machine translation. Technical report, JHU workshop 1999

[11] S. Vogel, H. Ney, C. Tillmann: Hmm-based word alignment in statistical translation. In: Proceedings of ACL. 1996

[12] F.J. Och, H. Ney: A systematic comparison of various statistical alignment models. Computational Linguistics 29(1) 2003

[13] Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistcial Society, series B 39(1) 1977

[14] M. Junczys-Dowmunt, Wprowadzenie do metod statystycznych w tłumaczeniu automatycznym, Investigationes Linguisticae, vol. XVI, 2008

[15] BerkleyAligner tool: http://code.google.com/p/berkeleyaligner/

[16] NATools: http://linguateca.di.uminho.pt/natools/

[17] T. Ildenfonso, G. Pereira Lopes, “Longest Sorted Sequence Algorithm for Parallel Text Alignment, Lecture Notes in Computer Science, 2005

[18] N. Varma, “Identifying Word Translations in Parallel Corpora Using Measures of Association, Department of Computer Science, University of Minnesota, 2002

[19] Qin Gao, Stephan Vogel, Parallel Implementation of Word Alignment Tool, 2008

[20] Qin Gao, Parallelizing the Training of Statistical Phrase-based Machine Translation

[21] Q. Gao, S. Vogel, Training Phrase-Nased Machine Translation Models on the Cloud – Open Source Machine Translation Toolkit Chaski, InterACT Lab, Carnegie University, USA, January 2010

[22] Tsuyoashi Okita, Data Cleaning for Word Alignment, CNGL / School of Computing, Dublin City University, Glasnevin

[23] System plików HDFS: http://hadoop.apache.org/hdfs/

[24] A. Szał, "The applications of statistical machine translation tool - PGIZA++", Proceeding of 41st Poznań Linguistic Meeting, 2010

[25] J. H. Clark, J. Weese, B. G. Ahn, A. Zollmann, Qin Gao, K. Heafield, A. Lavie, The Machine Translation Toolpack for LoonyBin: Automated Management of Experimental Machine Translation HyperWorkflows, January 2010

[26] J. H. Clark, J. Weese, B. G. Ahn, A. Zollmann, Qin Gao, K. Heafield, A. Lavie, The Machine Translation Toolpack for LoonyBin: Machine Translation and HyperWorkflows

[27] P. Koehn, H. Hoang, A. Birch, Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: Open source toolkit for statistical machine translation. In: ACL, ACL (2007)

[28] G. Doddongton, “Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics, National Institute of Standards and Technology, USA, 2002

[29] S. Banerjee, A. Lavie, "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments", Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43rd Annual Meeting of the Association of Computational Linguistics (ACL-2005), Michigan, 2005

[30] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul, "A Study of Translation Edit Rate with Targeted Human Annotation", Proceedings of Association for Machine Translation in the Americas, 2006

[31] A. Zollmann, A. Venugopal, “Syntax Augmented Machine Translation via Chart Parsing”, NAACL 2006 - Workshop on statistical machine translation, New York, 2006

[32] Heafield, Hanneman, Lavie, “Machine Translation System Combination with Flexible Word Ordering”, Proceedings of EACL 2009 Fourth Workshop on Statistical Machine Translation, Athens, Greece, 2009

[33] LoonyBin tool: http://www.cs.cmu.edu/~jhclark/loonybin

[34] P. Liang, B. Taskar, and D. Klein, “Alignment by agreement,” in Proceedings of ACL-COLING, pp. 104–111, 2006

[35] Och, Franz J. and Hermann Ney. A comparison of alignment models for statistical machine translation. In COLING ’00: The 18th International Conference on Computational Linguistics, pp. 1086–1090, Germany, August 2000

[36] Ewaluacja jakości tłumaczenia, http://pl.wikipedia.org/wiki/BLEU

[37] Papineni, K., Roukos, S., Ward, T., and Zhu, W. J. "BLEU: a method for automatic evaluation of machine translation" in ACL-2002: 40th Annual meeting of the Association for Computational Linguistics pp. 311–318, 2002.

[38] Callison-Burch, C., Osborne, M. and Koehn, P. (2006) "Re-evaluating the Role of BLEU in Machine Translation Research" in 11th Conference of the European Chapter of the Association for Computational Linguistics: EACL, pp. 249–256, 2006

[39] Denoual, E. and Lepage, Y., "BLEU in characters: towards automatic MT evaluation in languages without word delimiters" in Companion Volume to the Proceedings of the Second International Joint Conference on Natural Language Processing pp. 81–86, 2005

[40] Lee, A. and Przybocki, M., NIST 2005 machine translation evaluation official results, 2005

[41] R. Mihalcea, T. Pedersen, An evaluation exercise for word alignment, Proceedings of HLT-NAACL, 2003

[42] P. Liang, B. Taskar, D. Klein, Alignment by agreement, Proceedings of ACL-COLING, 2006

[43] J. DeNero and D. Klein, Tailoring word alignments to syntactic machine translation, Proceedings of ACL, 2007

[44] A. Szał, M. Junczys-Dowmunt, "SyMGiza++: A Tool for Parallel Computation of Symmetrized Word Alignment Models", Proceedings of 5th International Multiconference on Computer Science and Information Technology, 2010

[45] A. Szał, M. Junczys-Dowmunt, "SyMGiza++: Symmetrized Word Alignment Models for Statistical Machine Translation", Lecture Notes in Computer Science, 2011

[46] P. Koehn, Statistical significance tests for machine translation evaluation, EMNLP, 2004

[47] Xiaojun Lin, Xinhao Wang, Xihong Wu, NLMP System Description for the 2006 NIST MT Evaluation. NIST 2006 Machine Translation Evaluation, 2006

[48] Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John D. Lafferty, I. Dan Melamed, David Purdy, Franz J. Och, Noach A. Smith, David Yarowsky, Statistical Machine Translation. Final Report JHU Workshop, 1999

[49] Marcin Junczys-Dowmunt, It's all about the Trees: Towards a Hybrid Syntax-Based MT System, October 2009

[50] D. Jurafsky, J. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition, 2000

[51] M. Kay, M. Röscheisen, Text-translation alignment, Computational Linguistics, 1993

[52] K. Knight, A Statistical MT Tutorial Workbook, 1999

[53] C. D. Manning, H. Schütse, Foundations of Statistical Natural Language Processing, The MIT Press, 1999

[54] I. D. Melamed, Models of translational equivalence among words, Computational Linguistics, 2000

[55] H. Sommers, Bilingual parallel corpora and Language Engineering, 2001

Spis tabel

Spis rysunków

Dodatek A

Algorytm działania symetryzacji na plikach Giza++: t3, a3, d3

W tym dodatku zostaną przedstawione algorytmy symetryzacji zastosowane w narzędziu SyMGiza++. Algorytmy te są uruchamiane po każdej iteracji kolejnego modelu dopasowania wyrazów metodami statystycznymi.

Algorytm symetryzacji prawdopodobieństwa tłumaczenia

Po każdym modelu dopasowania wyrazów w narzędziu SyMGiza++ następuje symetryzacja plików t3. Pliki te opisują prawdopodobieństwa tłumaczenia poszczególnych wyrazów. Ich struktura składa się z następujących elementów:

- identyfikator wyrazu docelowego,
- identyfikator wyrazu źródłowego,
- prawdopodobieństwo tłumaczenia dla danej pary wyrazów.

Zaimplementowany algorytm symetryzacji prawdopodobieństw tłumaczenia został przedstawiony w tabeli 18. W pętli „while” dla każdej pary zdań jest pobierany wektor wyrazów źródłowych i docelowych. Następnie w podwójnej pętli „for” następuje przemieszczanie po poszczególnych wyrazach i pobranie prawdopodobieństwa tłumaczenia poszczególnych wyrazów z obu kierunków obliczeń. Program wylicza średnią arytmetyczną tych prawdopodobieństw i zapisuje w obu tablicach prawdopodobieństwa tłumaczenia. Ponadto zwiększając dzielnik prawdopodobieństwa, zmniejsza się prawdopodobieństwa tłumaczenia wyrazów niewystępujące w obu kierunkach.

Implementacja algorytmu symetryzacji prawdopodobieństw tłumaczenia

```
//_____symetrization of t files..._____
    sHandler1.rewind();
    sentPair sent ;
    WordIndex i, j, l, m ;
                                while(sHandler1.getNextSentence(sent)){
    <WordIndex>& es = sent.eSent;
    Vector<WordIndex>& fs = sent.fSent;
    const float so  = sent.getCount();
    l = es.size() - 1;
    m = fs.size() - 1;
```

```
        for(j=1; j <= m; j++)
        for(i = 0; i <= l; i++)
        {
                PROB pr1 = tTable.getProb(es[i],fs[j]);
                PROB pr2 = (*anotherModel1).tTable.getProb(fs[j],es[i]);
                if(pr1!=PROB_SMOOTH && pr2!=PROB_SMOOTH) {
                        PROB srednia = (pr1+pr2)/2;
                                tTable.insert(es[i],fs[j], 0.0, srednia);
                        (*anotherModel1).tTable.insert(fs[j],es[i], 0.0, srednia);

                }
                else{
                 if(pr1 > PROB_SMOOTH)
                                tTable.insert(es[i],fs[j], 0.0, pr1/m1TMultiplier);
                 if(pr2 > PROB_SMOOTH)
                                (*anotherModel1).tTable.insert(fs[j],es[i], 0.0, pr2/m1TMultiplier);

                 }
        }
    }

//_____________________________________________________________

    (*anotherModel1).symetrizeT=true;
```

Tabela 18. Implementacja algorytmu symetryzacji prawdopodobieństw tłumaczenia.

Algorytm symetryzacji prawdopodobieństwa dopasowania pozycji

Symetryzacja prawdopodobieństwa dopasowania pozycji (pliki a3) następuje po każdej iteracji modelu HMM, 3. i 4. Struktura pliku przechowującego te prawdopodobieństwa składa się z:

- pozycji wyrazu docelowego,
- pozycji wyrazu źródłowego,
- długości zdania źródłowego,
- długości zdania docelowego,
- prawdopodobieństwa, że wyraz źródłowy na danej pozycji ma swoje tłumaczenie w postaci wyrazu docelowego na danej pozycji dla pary zdań o podanej długości.

Implementacja algorytmu symetryzacji prawdopodobieństw dopasowania pozycji wyrazów została przedstawiona w tabeli 19. W pętli „while” są pobierane kolejne pary zdań

zawierające wektory z wyrazami źródłowymi i docelowymi. W podwójnej pętli „for" następuje odczyt prawdopodobieństw dopasowania pozycji wyrazów dla obu kierunków obliczeń. Jeżeli te prawdopodobieństwa mają wartość większą od zera, następuje obliczenie średniej wartości arytmetycznej z obu prawdopodobieństw i ich zapis do odpowiednich tabel.

Implementacja algorytmu symetryzacji prawdopodobieństw dopasowania pozycji

```
//_____symetrization o a files..._____

symNumber=0;//number of iteration after symetrization
while(((*anotherHMM).isEnd==false || (*anotherHMM).iterationNumber!=iterationNumber) && symetrizeT==false) {
sleep(1);
}
if(symetrizeT){
    symetrizeT=false;
} else{
   (*anotherHMM).isEnd=false;
   isEnd=false;

   sHandler1.rewind();
   sentPair sent ;
   WordIndex i, j, l, m ;
   while(sHandler1.getNextSentence(sent)){
      Vector<WordIndex>& es = sent.eSent;
      Vector<WordIndex>& fs = sent.fSent;
      const float so  = sent.getCount();
      l = es.size() - 1;
      m = fs.size() - 1;
      for(j=1; j <= m; j++)
            for(i = 0; i <= l; i++)
            {
            if(l==100 && m==100){
                  double a = aTable.getValue(i,j,100,100);
                  double b = (*anotherHMM).aTable.getValue(j,i,100,100);
                  if(a>0.000001 && b>0.000001) {
                        double sredniaA = (a+b)/2;
                        aTable.setValue(i,j, 100, 100,sredniaA) ;
                        (*anotherHMM).aTable.setValue(j,i, 100, 100,sredniaA);
                        }
```

```
                    }
                }
            }
            (*anotherHMM).symetrizeT=true;

    }
```

Tabela 19. Implementacja algorytmu symetryzacji prawdopodobieństw dopasowania pozycji.

Algorytm symetryzacji prawdopodobieństwa zniekształcenia

Symetryzacja prawdopodobieństwa zniekształcenia (pliki d3) następuje po każdej iteracji dla modelu 3. i 4. Struktura tego pliku jest taka sama jak pliku a3, z tą różnicą, że pozycje wyrazu źródłowego i docelowego są zamienione miejscami.

Implementacja algorytmu symetryzacji prawdopodobieństwa zniekształcenia wyrazów została przedstawiona w tabeli 20. Podobnie jak w poprzednich algorytmach następuje pobranie kolejnych par zdań w pętli „while". W podwójnej pętli następuje odczyt wartości prawdopodobieństw zniekształcenia, obliczenie średniej arytmetycznej tych prawdopodobieństw i zapis nowej wartości do odpowiednich tablic prawdopodobieństw.

Implementacja algorytmu symetryzacji prawdopodobieństw zniekształcenia

```
//_____symetrization of d files..._____
sHandler1.rewind();
sentPair sent ;
WordIndex i, j, l, m ;
while(sHandler1.getNextSentence(sent)){
    Vector<WordIndex>& es = sent.eSent;
    Vector<WordIndex>& fs = sent.fSent;
    const float so  = sent.getCount();
    l = es.size() - 1;
    m = fs.size() - 1;
    for(j=1; j <= m; j++)
        for(i = 0; i <= l; i++){
            double ad = dTable.getValue(i,j,100,100);
            double bd = (*anotherM3).dTable.getValue(j,i,100,100);
            if(ad>0.000001 && bd>0.000001) {
                double sredniaD = (ad+bd)/2;
                dTable.setValue(i,j, 100, 100,sredniaD) ;
                (*anotherM3).dTable.setValue(j,i, 100, 100,sredniaD);
            }
```

```
                }
    }
    (*anotherM3).symetrizeT=true;
```

Tabela 20. Implementacja algorytmu symetryzacji prawdopodobieństw zniekształcenia.

Algorytm działania symetryzacji na końcu obliczeń

Symetryzacja końcowa odbywa się za pomocą algorytmu *refined*. Został on dokładnie opisany w rozdziale 2.2.2. Można użyć również innych opisanych w powyższym rozdziale algorytmów symetryzacji. Kod źródłowy symetryzacji końcowej został przedstawiony w tabeli 21. Najważniejszym elementem jest funkcja „*last_symetrize(...)*” w której następuje odczyt plików wejściowych, wybór obpowiedniej metody symetryzacji i utworzenie plików wynikowych.

Implementacja algorytmu symetryzacji końcowej w programie SyMGiza++

```
    // global variables and constants

    int* fa; //counters of covered foreign positions
    int* ea; //counters of covered english positions
    int** A; //alignment matrix with information symmetric/direct/inverse alignments
    int verbose=0;
    //read an alignment pair from the input stream.
    int lc = 0;

    int getals(fstream& inp,int& m, int *a,int& n, int *b,ostream& out)
    {
      char w[MAX_WORD], dummy[10];
      string tgtsent;
      int i,j,freq;
      if (inp >> freq){
        ++lc;
        //target sentence
        inp >> n; assert(n<MAX_N);
        for (i=1;i<=n;i++){
          inp >> setw(MAX_WORD) >> w;
          if (strlen(w)>=MAX_WORD-1) {
            cerr << lc << ": target len=" << strlen(w) << " is not less than MAX_WORD-1=" << MAX_WORD-1
<< endl;
            assert(strlen(w)<MAX_WORD-1);
```

```
        }
            tgtsent+=w;
            tgtsent+=" ";
        }

        inp >> dummy; //# separator
        // inverse alignment
        for (i=1;i<=n;i++) inp >> b[i];
        //source sentence
        inp >> m; assert(m<MAX_M);
        for (j=1;j<=m;j++){
          inp >> setw(MAX_WORD) >> w;
          if (strlen(w)>=MAX_WORD-1) {
            cerr << lc << ": source len=" << strlen(w) << " is not less than MAX_WORD-1=" << MAX_WORD-1
<< endl;
            assert(strlen(w)<MAX_WORD-1);
          }
            out << w << " ";
        }
            out << "{##} " << tgtsent << "{##} ";
        inp >> dummy; //# separator
        // direct alignment
        for (j=1;j<=m;j++) {
          inp >> a[j];
          assert(0<=a[j] && a[j]<=n);
        }
        //check inverse alignemnt
        for (i=1;i<=n;i++)
          assert(0<=b[i] && b[i]<=m);
        return 1;
      }
      else
        return 0;
    };

    //compute union alignment
    int prunionalignment(fstream& out,int m,int *a,int n,int* b){
      ostringstream sout;
      for (int j=1;j<=m;j++)
        if (a[j])
```

```
      sout << j-1 << "-" << a[j]-1 << " ";
  for (int i=1;i<=n;i++)
    if (b[i] && a[b[i]]!=i)
      sout << b[i]-1 <<  "-" << i-1 << " ";
  //fix the last " "
  string str = sout.str();
  if (str.length() == 0)
    str = "\n";
  else
    str.replace(str.length()-1,1,"\n");
  out << str;
        out.flush();
        return 1;
}

//Compute intersection alignment
int printersect(fstream& out,int m,int *a,int n,int* b){
  ostringstream sout;
  for (int j=1;j<=m;j++)
    if (a[j] && b[a[j]]==j)
      sout << j-1 << "-" << a[j]-1 << " ";
  //fix the last " "
  string str = sout.str();
  if (str.length() == 0)
    str = "\n";
  else
    str.replace(str.length()-1,1,"\n");
  out << str;
        out.flush();
       return 1;
}

//Compute target-to-source alignment
int printtgttosrc(fstream& out,int m,int *a,int n,int* b){
  ostringstream sout;
  for (int i=1;i<=n;i++)
    if (b[i])
      sout << b[i]-1 << "-" << i-1 << " ";
  //fix the last " "
  string str = sout.str();
```

```
    if (str.length() == 0)
      str = "\n";
    else
      str.replace(str.length()-1,1,"\n");
    out << str;
          out.flush();
          return 1;
  }

  //Compute source-to-target alignment
  int printsrctotgt(fstream& out,int m,int *a,int n,int* b){
    ostringstream sout;
    for (int j=1;j<=m;j++)
      if (a[j])
        sout << j-1 << "-" << a[j]-1 << " ";
    //fix the last " "
    string str = sout.str();
    if (str.length() == 0)
      str = "\n";
    else
      str.replace(str.length()-1,1,"\n");
    out << str;
          out.flush();
         return 1;
  }

  //Compute Grow Diagonal Alignment
  //Nice property: you will never introduce more points
  //than the unionalignment alignemt. Hence, you will always be able
  //to represent the grow alignment as the unionalignment of a
  //directed and inverted alignment
  int printgrow(fstream& out,int m,int *a,int n,int* b, bool diagonal=false,bool final=false,bool
bothuncovered=false){
    ostringstream sout;
    vector <pair <int,int> > neighbors; //neighbors
    pair <int,int> entry;
    neighbors.push_back(make_pair(-1,-0));
    neighbors.push_back(make_pair(0,-1));
    neighbors.push_back(make_pair(1,0));
    neighbors.push_back(make_pair(0,1));
```

```
if (diagonal){
  neighbors.push_back(make_pair(-1,-1));
  neighbors.push_back(make_pair(-1,1));
  neighbors.push_back(make_pair(1,-1));
  neighbors.push_back(make_pair(1,1));
}
int i,j,o;
//covered foreign and english positions
 memset(fa,0,(m+1)*sizeof(int));
 memset(ea,0,(n+1)*sizeof(int));
//matrix to quickly check if one point is in the symmetric
//alignment (value=2), direct alignment (=1) and inverse alignment
for (int i=1;i<=n;i++) memset(A[i],0,(m+1)*sizeof(int));
set <pair <int,int> > currentpoints; //symmetric alignment
set <pair <int,int> > unionalignment; //union alignment
pair <int,int> point; //variable to store points
set<pair <int,int> >::const_iterator k; //iterator over sets
//fill in the alignments
for (j=1;j<=m;j++){
  if (a[j]){
    unionalignment.insert(make_pair(a[j],j));
    if (b[a[j]]==j){
      fa[j]=1;ea[a[j]]=1;
      A[a[j]][j]=2;
      currentpoints.insert(make_pair(a[j],j));
    }
    else
      A[a[j]][j]=-1;
  }
}
for (i=1;i<=n;i++)
  if (b[i] && a[b[i]]!=i){ //not intersection
    unionalignment.insert(make_pair(i,b[i]));
    A[i][b[i]]=1;
  }
int added=1;
while (added){
  added=0;
  ///scan the current alignment
  for (k=currentpoints.begin();k!=currentpoints.end();k++){
```

```
        //cout << "{"<< (k->second)-1 << "-" << (k->first)-1 << "}";
        for (o=0;o<neighbors.size();o++){
          //cout << "go over check all neighbors\n";
          point.first=k->first+neighbors[o].first;
          point.second=k->second+neighbors[o].second;
          //cout << point.second-1 << " " << point.first-1 << "\n";
          //check if neighbor is inside 'matrix'
          if (point.first>0 && point.first <=n && point.second>0 && point.second<=m)
            //check if neighbor is in the unionalignment alignment
            if (b[point.first]==point.second || a[point.second]==point.first){
              //cout << "In unionalignment ";cout.flush();
              //check if it connects at least one uncovered word
              if (!(ea[point.first] && fa[point.second]))
              {
                //insert point in currentpoints!
                currentpoints.insert(point);
                A[point.first][point.second]=2;
                ea[point.first]=1; fa[point.second]=1;
                added=1;
                //cout << "added grow: " << point.second-1 << "-" << point.first-1 << "\n";cout.flush();
              }
            }
        }
      }
    }
    if (final){
      for (k=unionalignment.begin();k!=unionalignment.end();k++)
        if (A[k->first][k->second]==1)
        {
          point.first=k->first;point.second=k->second;
          //one of the two words is not covered yet
          //cout << "{" << point.second-1 << "-" << point.first-1 << "} ";
          if ((bothuncovered &&  !ea[point.first] && !fa[point.second]) ||
            (!bothuncovered && !(ea[point.first] && fa[point.second])))
          {
            //add it!
            currentpoints.insert(point);
            A[point.first][point.second]=2;
            //keep track of new covered positions
            ea[point.first]=1;fa[point.second]=1;
```

```
                //added=1;
                //cout << "added final: " << point.second-1 << "-" << point.first-1 << "\n";
              }
            }
              for (k=unionalignment.begin();k!=unionalignment.end();k++)
                if (A[k->first][k->second]==-1)
                {
                  point.first=k->first;point.second=k->second;
                  //one of the two words is not covered yet
                  //cout << "{" << point.second-1 << "-" << point.first-1 << "} ";
                  if ((bothuncovered &&  !ea[point.first] && !fa[point.second]) ||
                      (!bothuncovered && !(ea[point.first] && fa[point.second])))
                  {
                    //add it!
                    currentpoints.insert(point);
                    A[point.first][point.second]=2;
                    //keep track of new covered positions
                    ea[point.first]=1;fa[point.second]=1;
                    //added=1;
                    //cout << "added final: " << point.second-1 << "-" << point.first-1 << "\n";
                  }
                }
        }
          for (k=currentpoints.begin();k!=currentpoints.end();k++)
            sout << k->second-1 << "-" << k->first-1 << " ";
          //fix the last " "
          string str = sout.str();
          if (str.length() == 0)
            str = "\n";
          else
            str.replace(str.length()-1,1,"\n");
          out << str;
          out.flush();
          return 1;
          return 1;
    }

    //Main file here
    int last_symetrize(int alignment, int diagonal, int final, int bothuncovered, const char* input, const char*
output){
```

```
    if (alignment==0){
      cerr << "usage: symal [-i=<inputfile>] [-o=<outputfile>] -a=[u|i|g] -d=[yes|no] -b=[yes|no] -f=[yes|no]
\n"
      << "Input file or std must be in .bal format (see script giza2bal.pl).\n";
      exit(1);
    }
    fstream inp(input,ios::in);
    fstream out(output,ios::out);
        if (!inp.is_open()){
                cerr << "cannot open " << input << "\n";
                exit(1);
        }
    if (!out.is_open()){
                cerr << "cannot open " << output << "\n";
                exit(1);
        }
    int a[MAX_M],b[MAX_N],m,n;
    fa=new int[MAX_M+1];
    ea=new int[MAX_N+1];
    int sents = 0;
    A=new int *[MAX_N+1];
    for (int i=1;i<=MAX_N;i++) A[i]=new int[MAX_M+1];
        switch (alignment){
                case UNION:
        cerr << "symal: computing union alignment\n";
                        while(getals(inp,m,a,n,b,out)) {
                  prunionalignment(out,m,a,n,b);
                  sents++;
                 }
                 cerr << "Sents: " << sents << endl;
                        break;
                case INTERSECT:
         cerr << "symal: computing intersect alignment\n";
                        while(getals(inp,m,a,n,b,out)) {
                  printersect(out,m,a,n,b);
                  sents++;
                 }
                 cerr << "Sents: " << sents << endl;
                        break;
      case GROW:
```

```
        cerr << "symal: computing grow alignment: diagonal ("
          << diagonal << ") final ("<< final << ")"
          <<  "both-uncovered (" << bothuncovered <<")\n";
          while(getals(inp,m,a,n,b,out))
              printgrow(out,m,a,n,b,diagonal,final,bothuncovered);

          break;
        case TGTTOSRC:
          cerr << "symal: computing target-to-source alignment\n";
          while(getals(inp,m,a,n,b,out)){
                printtgttosrc(out,m,a,n,b);
                sents++;
              }
              cerr << "Sents: " << sents << endl;
           break;
        case SRCTOTGT:
          cerr << "symal: computing source-to-target alignment\n";
          while(getals(inp,m,a,n,b,out)){
                printsrctotgt(out,m,a,n,b);
                sents++;
              }
              cerr << "Sents: " << sents << endl;
           break;
                default:
                        exit(1);
          }
      delete [] fa; delete [] ea;
      for (int i=1;i<=MAX_N;i++) delete [] A[i];
      delete [] A;
      exit(0);
    }
```

Tabela 21. Implementacja algorytmu symetryzacji końcowej w programie SyMGiza++.

Dodatek B

Instalacja i konfiguracja narzędzia SyMGiza++

Proces instalacji

Proces instalacji narzędzia SyMGiza++ wymaga podstawowej znajomości środowiska Linux. Przed przystąpieniem do instalacji należy zainstalować biblioteki boost, dostępne na stronie boost.org.

Po rozpakowaniu paczki instalacyjnej, proces instalacji wymaga wykonania następujących kroków:

./configure --prefix=installation_folder	- zdefiniowanie folderu instalacyjnego dla narzędzia SyMGiza++
make	- kompilacja plików źródłowych narzędzia
make install	- instalacja plików wykonywalnych

Parametry wejściowe i wyjściowe

SyMGiza++ bazuje na narzędziu MGiza++. W związku z tym większość parametrów jest taka sama w obu programach. Zmieniono jedynie parametry, które teraz odnoszą się do obliczeń dwukierunkowych. Ponadto dodano kilka parametrów procesu symetryzacji obliczeń.

Narzędzie SyMGiza++ można wywołać z następującymi parametrami:

- -ncpu – ilość wątków, parametr ten jest używany dla obu kierunków obliczeń, np. -ncpu 5 oznacza uruchomienie pięciu wątków dla każdego kierunku obliczeń, co daje w sumie dziesięć wątków
- pliki wejściowe – opis parametrów wejściowych:
    - –testcorpusfile (lub –tc) – plik korpusu wejściowego (plik z rozszerzeniem snt); plik ten jest wykorzystywany w procesie dopasowania wyrazów, ale obliczone wartości nie mają wpływu na modele,

- –sourcevocabularyfile (lub –s) – plik słownika źródłowego (z rozszerzeniem vcb),
- –targetvocabularyfile (lub –t) – plik słownika docelowego (z rozszerzeniem vcb),

- parametry określające liczbę iteracji poszczególnych modeli dopasowania tekstu:
  - –m1 (lub –noiterationsmodel1) – ilość iteracji dla modelu 1.,
  - –m2 (lub –noiterationsmodel2) – ilość iteracji dla modelu 2.,
  - –m3 (lub –noiterationsmodel3) – ilość iteracji dla modelu 3.,
  - –m4 (lub –noiterationsmodel4) – ilość iteracji dla modelu 4.,
  - –m5 (lub –noiterationsmodel5) – ilość iteracji dla modelu 5.,
  - –m6 (lub –noiterationsmodel6) – ilość iteracji dla modelu 6.,
  - –mh (lub –numberofiterationsforhmmalignmentmodel) – ilość iteracji dla modelu HMM,
- pliki wyjściowe – poniższe parametry opisują pliki wyjściowe utworzone przez narzędzie SyMGiza++ (w poniższych opisach n oznacza wartość parametru):
  - –compactalignmentformat true/false – skompresowany format prezentujący powiązania poprzez linki np. 1 2 3 2,
  - –countoutputprefix – parametr mówiący narzędziu, że ma zwrócić tablice obliczeń przed normalizacją, zamiast znormalizowanego modelu,
  - -dumpcount true/false – zrzut obliczeń dla modelu znormalizowanego; obliczenia te są zrzucane w końcowym kroku obliczeń,
  - –dumpcountusingwordstring true/false – zrzut obliczeń z użyciem słów zamiast identyfikatorów,
  - –logfile (lub –l) – zrzut logów narzędzia do pliku,
  - –model1dumfrequency (lub –t1) – zrzut tablic wynikowych modelu 1. po n iteracjach; wartość jeden oznacza zrzut wyników po każdej iteracji modelu 1.,
  - –model2dumfrequency (lub –t2) – zrzut wyników dla modelu 2. po n iteracjach,
  - –model345dumfrequency (lub –t345) – zrzut wyników dla modelu 3., 4. i 5. po n iteracjach, dla wartości 1 zrzut nastąpi po każdej iteracji modelu 3., 4. i 5.,
  - –hmmdumfrequency (lub –th) – zrzut wyników dla modelu HMM po n iteracjach,

    - –transferdumpfrequency (lub –t2to3) – zrzut wyników dodatkowej iteracji będącej konwersją z modelu 2. do 3.,
    - –nbestalignmets true/false – zrzut n najlepszych dopasowań dla plików modeli i plików dopasowań,
    - –onlyaldumps true/false – tylko zrzut dopasowań,
- inne parametry:
    - –probsmooth float – w przypadku braku prawdopodobieństw dla aktualnego modelu zostanie użyta podana w tym parametrze wartość.

Następujące nowe pramatery zostały zdefiniowne dla rozszerzenia SyMGiza++:

- –corpusfileEF (lub –cef) – plik wejściowy korpusu dla kierunku pierwszego (dopasowanie z pliku źródłowego do docelowego) – plik z rozszerzeniem snt,

- -corpusfileFE (lub –cfe) – plik wejściowy dla kierunku drugiego (dopasowanie z pliku docelowego do źródłowego) – plik z rozszerzeniem snt,
- -CoocurrenceFileFE – tablica współwystępowań wyrazów dla pierwszego kierunku; tablica ta pozwala przyśpieszyć obliczenia,
- -CoocurrenceFileEF - tablica współwystępowań wyrazów dla drugiego kierunku,
- -oef – alias dla wygenerowanych plików wyjściowych w pierwszym kierunku obliczeń; z takim aliasem będą wszystkie pliki wygenerowane w modelach dla tego kierunku obliczeń
- -ofe – alias dla wygenerowanych plików wyjściowych w drugim kierunku obliczeń,
- -o – alias dla wygenerowanych plików wyjściowych wspólnych dla obu kierunków obliczeń,
- -m1symfrequency – częstość symetryzacji obliczeń dla modelu 1.; parametr ten określa, jak często ma być wykonywana symetryzacja w modelu 1.; wartość 1 oznacza, że symetryzacja będzie wykonywana po każdej iteracji modelu 1.,
- -m2symfrequency - częstość symetryzacji obliczeń dla modelu 2.,
- -m345symfrequency - częstość symetryzacji obliczeń dla modelu 3., 4. i 5.,
- -mhsymfrequency - częstość symetryzacji obliczeń dla modelu HMM,
- -tm – mnożnik tablic t (prawdopodobieństwa tłumaczenia), parametr ten określa wagę danej symetryzacji; dla większej wartości parametru słowa powiązane w obu kierunkach obliczeń otrzymują większe prawdopodobieństwo dobrego tłumaczenia,

- -es – parametr określający, czy ma zostać wykonana symetryzacja końcowa (1 - tak, 0 - nie), a jeśli tak to z jakimi parametrami:
    - -alig – typ symetryzacji końcowej; dostępne opcje to (podobnie jak w dopasowaniu za pomocą narzędzia Moses):
        - intersect
        - union
        - grow
        - srctotgt
        - tgttosrc
    - -diagonal – czy wykonać diagonalizacje na symetryzacji końcowej (yes – tak lub no - nie),
    - -final - (yes - tak lub no - nie),
    - -both - (yes - tak lub no - nie).

Wyjściowe dopasowanie wyrazów dla narzędzia SyMGiza++ składa się plików wygenerowanych z każdego procesu obliczeń:

- dla pierwszego kierunku:
    - prefixOfDirectionOne.A3.final.part0,
    - prefixOfDirectionOne.A3.final.part1,
    - ...,
    - prefixOfDirectionOne.A3.final.partN,
- dla drugiego kierunku:
    - prefixOfDirectionTwo.A3.final.part0,
    - prefixOfDirectionTwo.A3.final.part1,
    - ...,
    - prefixOfDirectionTwo.A3.final.partN.

Aby powiązać pliki z każdego kierunku należy wykonać dwa polecenia:

*installation_folder/scripts/merge_alignment.py prefixOfDirectionOne.A3.final.part** > *prefixOfDirectionOne.A3.final*

*installation_folder/scripts/merge_alignment.py prefixOfDirectionTwo.A3.final.part** > *prefixOfDirectionOne.A3.final*

W narzędziu SyMGiza++ powyższy krok wiązania obliczeń jest wykonywany automatycznie.

Przykład uruchomienia narzędzia

Poniższy przykład ukazuje kroki dopasowania wyrazów w korpusach dwujęzycznych z użyciem narzędzia SyMGiza++:

- komenda plain2snt generuje pliki snt dla każdego kierunku obliczeń z dwóch korpusów (Corpus1, Corpus2):

  installation_folder/src/plain2snt ./Corpus1 ./$Corpus2
  installation_folder/src/plain2snt ./Corpus2 ./$Corpus1

- komenda mkcl pozwala wygenerować klasy słów (ang. words classes):

  installation_folder/src/mkcls/mkcls –pCorpus1 –VCorpus1.vcb.classes
  installation_folder/src/mkcls/mkcls -pCorpus2 -VCorpus2.vcb.classes

- komenda snt2cooc generuje pliki współwystępowania (ang. co-occurrence files):

installation_folder/src/snt2cooc c1-c2.cooc Corpus1.vcb Corpus2.vcb Corpus1_Corpus2.snt
installation_folder/src/snt2cooc c2-c1.cooc Corpus2.vcb Corpus1.vcb Corpus2_Corpus1.snt

Powyższe komendy pozwalają wygenerować wszystkie niezbędne pliki wejściowe do głównego procesu obliczeń wykonywanego za pomocą narzędzia SyMGiza++:

installation_folder /src/mgiza -ncpus 2 -s Corpus1.vcb -t Corpus2.vcb
–cef Corpus1_Corpus2.snt -cfe Corpus2_Corpus1.snt -oef c1 -ofe c2 -o all
-CoocurrenceFileEF c1-c2.cooc -CoocurrenceFileFE c2-c1.cooc
-m1 5 -m2 5 -m3 5 -m4 5 -mh 5 -t1 5 -t2 5 -th 5 -t345 5
-m1symfrequency 5 -m2symfrequency 5 -mhsymfrequency 5 -m345symfrequency 5
-tm 2 -es 1 -alig intersect -diagonal yes -final yes -both yes

Dla danego przypadku mamy zdefiniowane następujące parametry programu:

- -npus 2 – dwa wątki dla każdego kierunku obliczeń

- -s Corpus1.vcb – nazwa korpusu wejściowego
- -t Corpus2.vcb – nazwa korpusu wyjściowego
- -cef Corpus2_Corpus1.snt – nazwa pliku snt dla pierwszego kierunku dopasowania
- -cfe Corpus2_Corpus1.snt – nazwa pliku snt dla drugiego kierunku dopasowania
- -oef c1 – alias c1 dla plików wyjściowych pierwszego kierunku dopasowania
- -ofe c2 – alias c2 dla plików wyjściowych drugiego kierunku dopasowania
- -o all – alias all dla wspólnych plików wyjściowych z obu kierunków dopasowania
- -CoocurrenceFileEF c1-c2.cooc – plik współwystępowania dla pierwszego kierunku obliczeń
- -CoocurrenceFileFE c2-c1.cooc – plik współwystępowania dla drugiego kierunku obliczeń
- -m1 5 – 5 iteracji dla modelu 1.
- -m2 5 – 5 iteracji dla modelu 2.
- -m3 5 – 5 iteracji dla modelu 3.
- -m4 5 – 5 iteracji dla modelu 4.
- -mh 5 – 5 iteracji dla modelu HMM
- -t1 5 – zrzut prawdopodobieństw dla modelu 1. po pięciu iteracjach tego modelu
- -t2 5 – zrzut prawdopodobieństw dla modelu 2. po pięciu iteracjach tego modelu
- -th 5 – zrzut prawdopodobieństw dla modelu HMM po pięciu iteracjach tego modelu
- -t345 5 – zrzut prawdopodobieństw dla modelu 3., 4. i 5. po pięciu iteracjach tego modelu
- -m1symfrequency 5 – symetryzacja po pięciu iteracjach modelu 1.
- -m2symfrequency 5 – symetryzacja po pięciu iteracjach modelu 2.
- -mhsymfrequency 5 – symetryzacja po pięciu iteracjach modelu HMM
- -m345symfrequency 5 – symetryzacja po pięciu iteracjach modelu 3., 4. i 5.
- -tm 2 – mnożnik 2 dla tablicy t prawdopodobieństwa tłumaczenia po symetryzacji
- -es 1 – uruchom symetryzację końcową
- -alig intersect – uruchom symetryzację końcową, za pomocą metody iloczynów
- -diagonal yes
- -final yes
- -both yes

Jako rezultat powyższej komendy otrzymujemy trzy istotne pliki dopasowania wyrazów:

- c1.A3.final – końcowa macierz dopasowania wyrazów dla obliczeń w kierunku korpus źródłowy – korpus docelowy bez symetryzacji końcowej,
- c2.A3.final – końcowa macierz dopasowania wyrazów dla obliczeń w kierunku korpus docelowy – korpus źródłowy bez symetryzacji końcowej,
- all.A3.final_symal – dopasowanie końcowe z symetryzacją końcową.

Narzędzie SyMGiza++ można pobrać ze strony **psi.amu.edu.pl.** Jest dostępne w zakładce „**Narzędzia do pobrania”.**

Aby ułatwić proces tworzenia dopasowań wyrazów z użyciem SyMGIZA++ utworzono skrypt uruchomieniowy bash. Umożliwia on uruchomienie narzędzia z domyślnymi ustawieniami, a także pozwala na modyfikację konfiguracji aplikacji. Aby uruchomić dany skrypt należy wykonać następującą komendę:

 # ./run.sh nazwa_pliku_źródłowego_korpusu nazwa_pliku_wyjściowego_korpusu ścieżka_zapisu_plików_wyjściowych