UNIWERSYTET im. ADAMA MICKIEWICZA
w POZNANIU
WYDZIAŁ MATEMATYKI I INFORMATYKI
KIERUNEK INFORMATYKA

Studia stacjonarne jednolite magisterskie

RAFAŁ JAWORSKI

TŁUMACZENIE TEKSTÓW PRAWNICZYCH PRZEZ ANALOGIE

Z wykorzystaniem technik rozpoznawania jednostek nazwanych (*NE*) oraz metod słownikowych.

PRACA MAGISTERSKA

PROMOTOR
dr hab. Krzysztof Jassem

Poznań, 2009

mojej najdroższej Agnieszce

OŚWIADCZENIE

Ja, niżej podpisany **Rafał Jaworski,** student Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczam, że przedkładaną pracę dyplomową pt:

„Tłumaczenie tekstów prawniczych przez analogie”

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 części innym osobom, ani nie odpisywałem tej rozprawy lub jej części od innych osób.

Oświadczam również, że egzemplarz pracy dyplomowej w formie wydruku komputerowego jest zgodny z egzemplarzem pracy dyplomowej w formie elektronicznej.

Jednocześnie przyjmuję do wiadomości, że gdyby powyższe oświadczenie okazało się nieprawdziwe, decyzja o wydaniu mi dyplomu zostanie cofnięta.

……..

SPIS TREŚCI

WSTĘP

Niniejsza praca traktuje o wybranej technice tłumaczenia automatycznego – tłumaczeniu przez analogie. Jest to technika, która ma szansę poprawić tłumaczenie automatyczne, którego jakość w obecnych czasach na ogół nie jest zadowalająca.

Tłumaczenie przez analogie znajduje zastosowanie przede wszystkim w tzw. „systemach wspomagających tłumaczenie". Są to systemy przeznaczone dla tłumaczy, które mają pomóc człowiekowi w wykonaniu tłumaczenia, a nie wykonać tłumaczenie samodzielnie. Skuteczne tłumaczenie przez analogie może znacząco skrócić czas pracy tłumacza nad przetłumaczeniem tekstu.

W niniejszej pracy szczególny nacisk został położony na tłumaczenie tekstów prawniczych. Są one szczególnie interesujące z punktu widzenia tłumaczenia automatycznego. Bogactwo języka prawniczego skłania do opracowywania specjalistycznych zasobów słownikowych oraz algorytmów przetwarzania tego języka. Jego jednoznaczność pozwala natomiast uzyskać stosunkowo wysoką jakość tłumaczenia automatycznego.

Praca dzieli się na część badawczą (rozdziały 1,2,3 i 4) oraz praktyczną (rozdziały 5,6 i 7). Część badawcza przybliża ideę tłumaczenia przez analogie oraz prezentuje dotychczasowe osiągnięcia na tym polu. Część praktyczna natomiast stanowi opis systemu NeLex – systemu tłumaczenia przez analogie, zaimplementowanego w ramach niniejszej pracy magisterskiej.

Pierwszy rozdział wyjaśnia najogólniej mechanizm tłumaczenia przez analogie. Rozdział drugi porównuje tłumaczenie przez analogie z tłumaczeniem przez transfer oraz opisuje operacje konieczne do wykonania przed przeprowadzeniem tłumaczenia przez analogie. W trzecim rozdziale omówione są dwa istniejące już systemy tłumaczenia przez analogie. Natomiast rozdział czwarty opisuje system heTMan – system tłumaczenia i zarządzania pamięcią tłumaczeń, który będzie korzystał z NeLex'a.

Najistotniejszym rozdziałem pracy jest rozdział piąty, który szczegółowo opisuje algorytm tłumaczenia przez analogie, wykorzystywany w systemie NeLex. Rozdział szósty przybliża te części algorytmu tłumaczącego systemu NeLex, które są specjalnie dostosowane do tłumaczenia tekstów prawniczych. W ostatnim rozdziale można zapoznać się z wynikami testów systemu NeLex. Rozdział ten prezentuje również możliwe drogi rozwoju systemu.

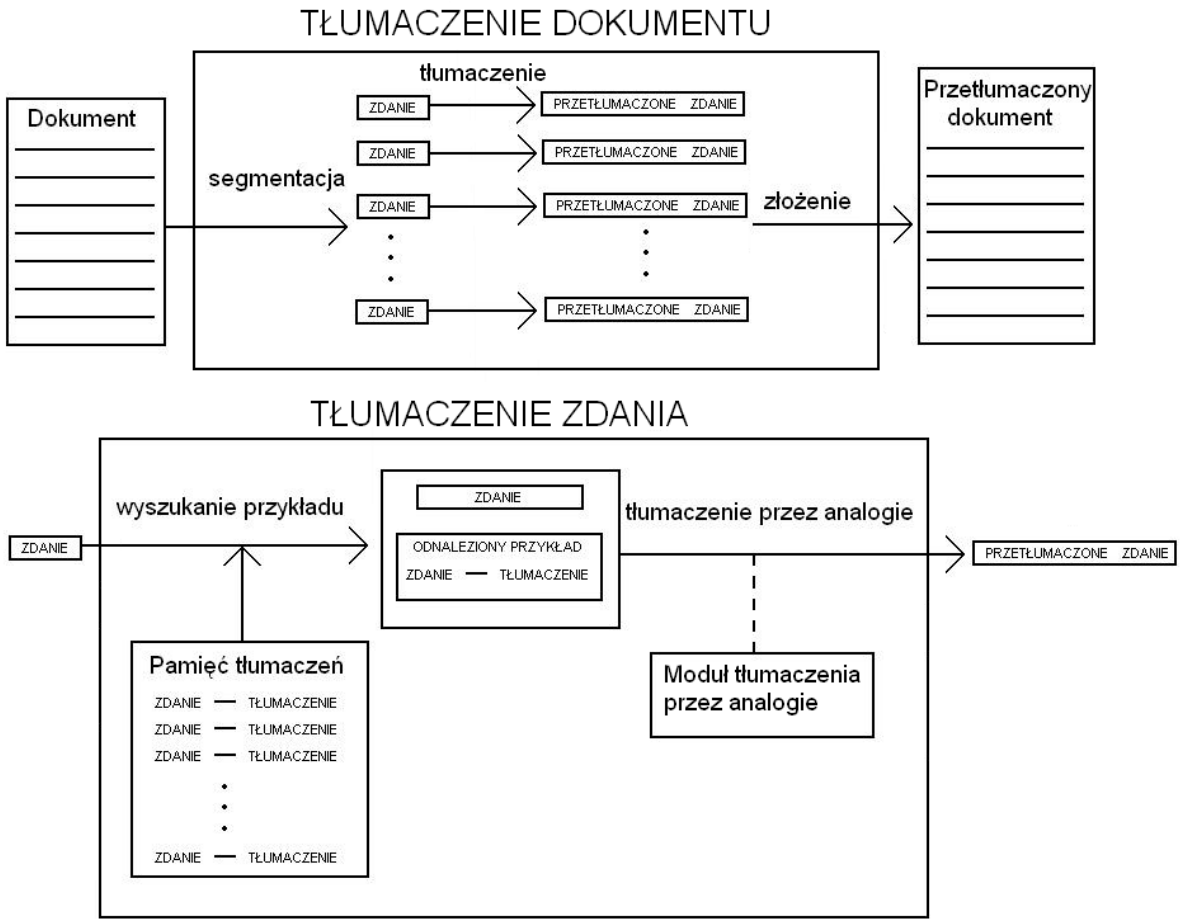
ROZDZIAŁ 1
ZAGADNIENIA WSTĘPNE

1.1. OGÓLNY ZARYS IDEI TŁUMACZENIA PRZEZ ANALOGIE

Tłumaczenie przez analogie opiera się na wykorzystaniu wcześniej opracowanych tłumaczeń w tłumaczeniu nowych tekstów. Zakładamy, że te istniejące tłumaczenia są dobrej jakości (najczęściej jest to równoznaczne z tym, że są wynikiem pracy człowieka). Dzięki nim nie ma konieczności prowadzenia głębokiej analizy gramatycznej tekstu do przetłumaczenia. Nie ma też potrzeby budowania skomplikowanych reguł tłumaczenia. To przykładowe tłumaczenia, traktowane jako wzorzec, wyznaczają reguły translacji.

Proces tłumaczenia przez analogie składa się z kilku etapów. Ponieważ w tym podejściu tłumaczy się tekst zdanie po zdaniu, musi on najpierw zostać podzielony na zdania. Operacja ta jest nazywana **segmentacją** (zostanie opisana bardziej szczegółowo w rozdziale 2.1.2). Niekiedy stosuje się dalszy podział – zdania dzieli się na frazy, (jak w systemie tłumaczenia przez analogie opisanym w artykule [Nagao, 1984]). Po segmentacji, na każdym z otrzymanych zdań (lub fraz) należy zastosować algorytm tłumaczący. Ostatnim krokiem jest złożenie przetłumaczonych zdań

i ewentualne przetwarzanie końcowe (stosuje się je w systemie tłumaczącym opracowanym przez Chińską Akademię nauk [CAS, 2004]). Etapy tłumaczenia są przedstawione na schemacie S1:



Schemat S1 – tłumaczenie dokumentu przez analogie

Moduł tłumaczenia przez analogie działa według algorytmu, przedstawionego w następnym rozdziale.

1.1.1. Algorytm tłumaczenia przez analogie

Założenia:

Zakładamy, że dysponujemy dużej wielkości zbiorem **T**, par: **(z, t)**, gdzie **z** to dowolne zdanie w języku naturalnym, a **t** jest tłumaczeniem zdania **z** na inny język naturalny. Na wejściu algorytm otrzymuje zdanie **w**, które jest w tym samym języku, co zdanie **z**. Zadaniem systemu tłumaczącego jest przetłumaczenie zdania **w** na język, w którym jest zdanie **t**.

Algorytm:

Działanie systemu tłumaczącego opisuje ogólny algorytm:

1. Znajdź taką parę **(z*, t*)**, że **z*** jest najbardziej podobne do **w,** spośród wszystkich par, należących do **T**.
2. Przetwórz **z*** tak, aby **z* = w**, zapisując listę **D** – operacji do tego koniecznych
3. Dla każdej operacji z **D**:
    a. Wykonaj operację na zdaniu **t***, o ile jest to możliwe
4. Zwróć **t*** jako wynik tłumaczenia
5. Podaj ocenę tłumaczenia (liczbę rzeczywistą z zakresu [0,1])

W niniejszej pracy nacisk zostanie położony na kroki 2-5 powyższego algorytmu. Wejściem do opracowywanego algorytmu będzie para zdań **(z*, t*)**, odszukana w zbiorze **T** przez inny – niezależny moduł – oraz zdanie do przetłumaczenia - **w**. Na wyjściu algorytm będzie zwracał przetłumaczone zdanie, wraz z oceną tłumaczenia.

1.2. SŁOWNIK WYKORZYSTYWANYCH POJĘĆ

Poniżej przedstawiony zostanie słownik wszystkich wykorzystywanych w tej pracy pojęć, z zakresu tłumaczenia przez analogie. Pomoże on sprawniej formułować wykorzystywane teorie. Większość pojęć odpowiada znaczeniowo analogicznym pojęciom z literatury. Część z nich została jednak celowo dospecyfikowana na potrzeby niniejszej pracy.

Tłumaczenie przez analogie – technika tłumaczenia automatycznego, polegająca na wykorzystaniu **Pamięci tłumaczeń.**
EBMT – skrót od angielskich słów ***E**xample-**B**ased **M**achine **T**ranslation*, oznaczających **Tłumaczenie przez analogie** (dosłownie: tłumaczenie oparte na przykładach).
Ocena tłumaczenia (w tłumaczeniu przez analogie) – liczba rzeczywista z przedziału [0,1], która jest miarą tego, w jakim stopniu powiodło się tłumaczenie zdania przez analogie. Ocenę tę podaje sam system, który wykonuje takie tłumaczenie. Wartość 1 oznacza pewność poprawności tłumaczenia, a 0 – niepowodzenie tłumaczenia danego zdania przez analogie.
Pamięć tłumaczeń – baza zawierająca wcześniej przetłumaczone zdania, jej elementami są **przykłady**.
Przykład – para: **zdanie źródłowe**, **zdanie docelowe**.

Zdanie – część tekstu, wydzielona przez algorytm segmentacji według pewnych zadanych reguł (zwanych **regułami segmentacji**).

Zdanie źródłowe – pierwszy element pary, będącej **przykładem**.

Zdanie docelowe – drugi element pary, będącej **przykładem** (tłumaczenie **zdania źródłowego).**

Język źródłowy – język, z którego tłumaczymy (**zdania źródłowe** są napisane w tym języku)

Język docelowy – język, na który tłumaczymy (**zdania docelowe** są napisane w tym języku)

Zdanie wejściowe – zdanie w **języku źródłowym**, które ma zostać przetłumaczone na **język docelowy.**

Słownik – plik zawierający informacje o tym, z jakiego **leksemu** pochodzi dane słowo oraz na jaki **leksem** może zostać przetłumaczone.

Leksem – zbiór słów wywodzących się od jednego, podstawowego słowa, a różniących się jedynie formą gramatyczną. Inaczej: zbiór **słowoform**.

Słowoforma – słowo w konkretnej formie gramatycznej.

Jednostka leksykalna – ciąg **tokenów**, który posiada jakieś znaczenie, to jest wpływa na sens zdania (np. **słowo**, **znak interpunkcyjny**, ale nie np. spacja)

Token – najmniejszy wydzielony ze zdania ciąg znaków, które są do siebie podobne (np. ciąg liter lub ciąg białych znaków)

Słowo – typ jednostki leksykalnej, ciąg liter

Jednostka nazwana (*NE – Named Entity*) – wyróżniony typ jednostki leksykalnej, jednostka obdarzona specjalnym znaczeniem

Znak interpunkcyjny – typ jednostki leksykalnej, jeden z następujących: .,!?()

Biały znak – znak niedrukowalny, na przykład spacja, tabulacja, nowa linia

Siatka odpowiedniości – schemat powiązań między jednostkami leksykalnymi dwóch zdań

Tablica odpowiedniości – tablicowy zapis **siatki odpowiedniości**

Segmentacja – operacja podziału tekstu na zdania

Reguła podziału (reguła segmentacji) – reguła definiująca miejsce w tekście, w którym ma nastąpić podział.

Reguła zabraniająca podziału (wyjątek) – reguła definiująca miejsce w tekście, w którym podział na pewno nie nastąpi.

Wyrażenie regularne – wzorzec napisu, zapisany w specjalnie do tego służącym języku. Na przykład wzorzec *„ab.*”* oznacza napis rozpoczynający się znakami *„ab”*, po których następuje dowolny inny znak, dowolną liczbę razy

Konkatenacja - złączenie tekstów.

1.3. PRZYKŁADY TŁUMACZENIA PRZEZ ANALOGIE

W rozdziale tym zostaną zaprezentowane proste przykłady tłumaczeń przez analogie. W przykładach tych będzie wykorzystywany algorytm systemu Ne-Lex, który zostanie dokładnie omówiony w rozdziale 5.

1.3.1. Przykład 1

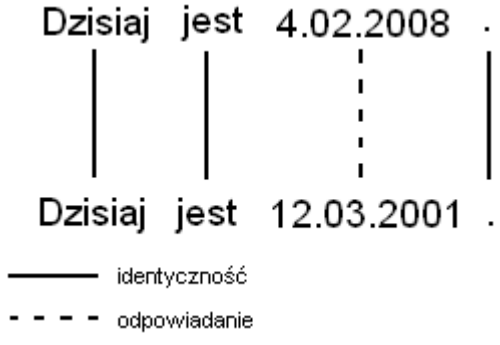
Pobrany z pamięci tłumaczeń przykład jest następujący:

pl: „Dzisiaj jest 12.03.2001.”
en: „Today is 2001/03/12.”

Zdanie wejściowe do przetłumaczenia:

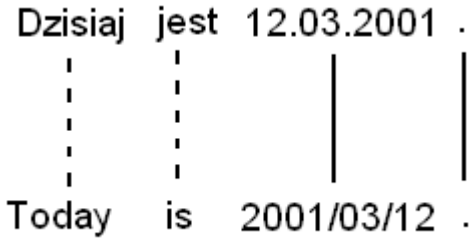
pl: „Dzisiaj jest 4.02.2008.”

Zostaje zbudowana siatka odpowiedniości miedzy zdaniem wejściowym, a źródłowym:



Schemat S2 – siatka odpowiedniości pomiędzy zdaniem wejściowym a zdaniem źródłowym z Pamięci Tłumaczeń

Dzięki siatce odpowiedniości widoczne jest, że zdanie wejściowe różni się od zdania źródłowego jedynie datą. Zostaje teraz zbudowana siatka odpowiedniości pomiędzy zdaniem źródłowym, a docelowym:



Schemat S3 – siatka odpowiedniości pomiędzy zdaniem źródłowym, a odpowiadającym mu zdaniem docelowym

Tłumaczenie wykorzystuje informacje przedstawione na schematach S2 i S3. Najpierw data 4.02.2008 zostaje przeniesiona ze zdania wejściowego do źródłowego (dzięki siatce odpowiedniości ze schematu S2 wiadomo, że można takiej operacji dokonać). Zdanie źródłowe zostaje przekształcone do postaci:

pl: „Dzisiaj jest 4.02.2008”

Ta sama data zostaje przeniesiona dalej ze zdania źródłowego do docelowego. Z siatki odpowiedniości ze schematu S3 zostaje odczytana informacja, w które miejsce należy wstawić tę datę w zdaniu docelowym. Musi ona być wstawiona w miejsce daty 2001/03/12. Dodatkowo, podczas przenoszenia zmieniony zostaje format daty na zgodny z językiem docelowym (angielski ze Stanów Zjednoczonych). Wynikiem tych operacji jest zdanie:

en: „Today is 2008/02/04.”

I to właśnie zdanie jest zwracane wynik tłumaczenia przez analogie. Takie tłumaczenie może być uznane za sukces, ponieważ udało się zniwelować jedyną różnicę pomiędzy zdaniem wejściowym, a źródłowym. Dlatego ocena tego tłumaczenia będzie bliska 100%.

1.3.2. Przykład 2

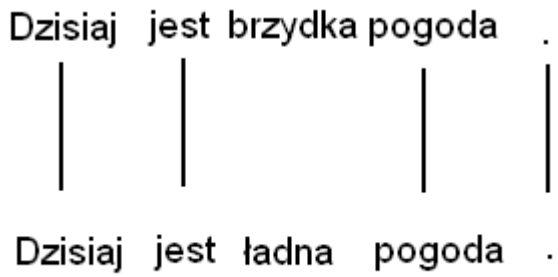
Tym razem zdania z przykładu są następujące:

pl: „Dzisiaj jest ładna pogoda.”
en: „The weather is nice today.”

Zdanie wejściowe:

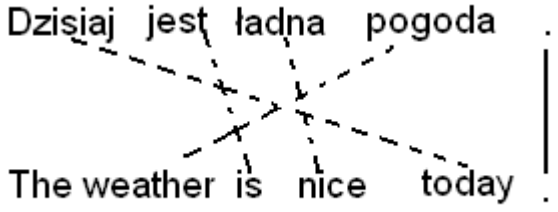
pl: „Dzisiaj jest brzydka pogoda.”

Ponownie zostaje zbudowana siatka odpowiedniości:



Schemat S4 – siatka odpowiedniości między zdaniem wejściowym, a docelowym

Widać, że zdanie wejściowe od źródłowego różni się tylko jednym wyrazem – w miejsce wyrazu *„ładna”* ze zdania źródłowego w zdaniu wejściowym występuje wyraz *„brzydka”*. Widoczne jest również, że słowo „brzydka” nie odpowiada żadnemu słowu ze zdania źródłowego oraz że słowo „ładna” nie odpowiada żadnemu słowu ze zdania wejściowego. Na tej podstawie zostaje podjęta decyzja, że słowo „ładna” może zostać zamienione słowem „brzydka”.
Zbudujmy teraz siatkę odpowiedniości między zdaniem źródłowym, a docelowym:



Schemat S5 – siatka odpowiedniości pomiędzy zdaniem źródłowym, a docelowym

Z niej pochodzi informacja, że słowo „ładna” ze zdania źródłowego może odpowiadać słowu „nice” ze zdania docelowego.

Następuje tłumaczenie: na podstawie informacji ze schematu S4, słowo „ładna” w zdaniu źródłowym zostaje zamienione słowem „brzydka”:

pl: „Dzisiaj jest brzydka pogoda.”

Ponieważ w zdaniu źródłowym zostało zamienione słowo „ładna”, wobec czego w zdaniu docelowym musi zostać zamienić słowo odpowiadające temu słowu. Ze schematu S5 wynika, że jest to słowo „nice”. W związku z tym, do zdania docelowego zostaje wstawione przetłumaczone na angielski słowo „brzydka”, w miejsce słowa „nice”:

en: „The weather is ugly today.”

To tłumaczenie wyszło dość dobrze. Jednak zwykle nie ma pewności, że operacja zamiany słowa powiedzie się. Dzieje się tak dlatego, że tłumaczenie pojedynczego słowa opiera się tylko na tłumaczeniu leksemów. Algorytm zaniedbuje to, czy podaje odpowiednią słowoformę tłumaczonego leksemu. Dla podobnych zdań, jak powyżej, mógłby on na przykład podać tłumaczenie:

en: „The weather is ugliest today.”

Błąd polegałby na użyciu niewłaściwej słowoformy leksemu. Nie jest to jednak jedyne źródło problemów. Najczęściej jedno słowo ma kilka znaczeń, czego nie uwzględnia algorytm. Z tego powodu możliwe jest na przykład otrzymanie tłumaczenia:

en: „The weather is naughty today.”

Mając to na uwadze, algorytm podaje niższą ocenę tłumaczenia, niż w przykładzie 1.

1.3.3. Przykład 3

Tym razem przykład wygląda następująco:

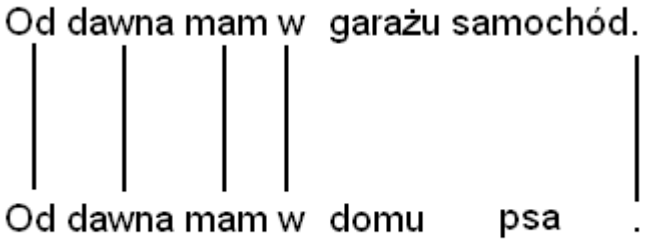
pl: „Od dawna mam w domu psa.”

en: „I have had a dog in the house for a long time.”

Zdanie wejściowe:

pl: „Od dawna mam w garażu samochód.”

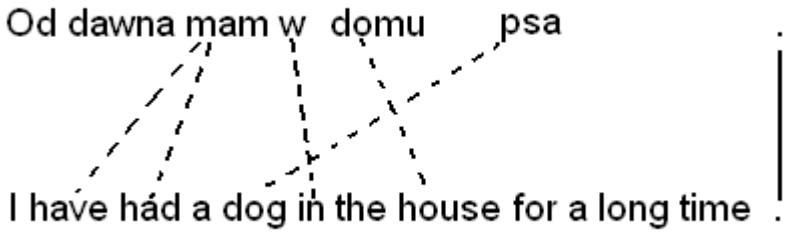
Siatka odpowiedniości między zdaniem wejściowym, a źródłowym:



Schemat S6 – siatka odpowiedniości

Tym razem w obu zdaniach łącznie występują aż 4 słowa, które nie odpowiadają żadnym innym. Nie zagłębiając się w analizę gramatyczną, w uproszczony sposób przyjmuje się, że słowo „garażu” będzie mogło być zamienione słowem „domu”, a kolejno słowo „samochód” – słowem „psa”.

Zostaje zbudowana siatka odpowiedniości pomiędzy zdaniem źródłowym, a docelowym:



Schemat S7 – siatka odpowiedniości pomiędzy zdaniem wejściowym, a źródłowym, o większym (niż w poprzednich przykładach) stopniu skomplikowania

Analogicznie do mechanizmu przedstawionego w przykładzie 2, zostaje najpierw podstawione w zdaniu źródłowym słowo „domu” słowem „garażu”, a słowo „psa” słowem „samochód”:

pl: „Od dawna mam w garażu samochód.”

Następnie słowa te są przenoszone do zdania docelowego, z wykorzystaniem z informacji ze schematu S7:

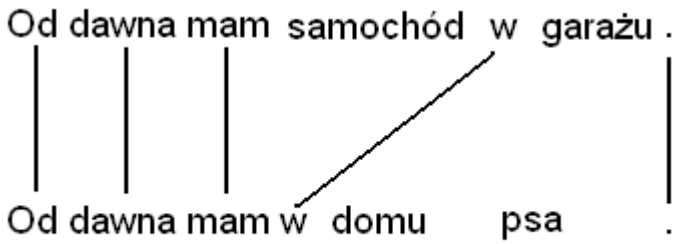
en: „I have had a car in the garage for a long time.”

Takie tłumaczenie ponownie może zostać uznać za poprawne. Jednak, podobnie jak w przykładzie 2, przy tłumaczeniu mogą wystąpić komplikacje, które obniżą jego jakość. Z tego powodu obniżeniu ulegnie ocena tłumaczenia, przy czym obniżenie to będzie jeszcze większe niż w przykładzie 2, gdyż teraz zostały podstawione dwa słowa.

Występuje jednak jeszcze jeden problem, wynikający z podstawiania więcej niż jednego słowa. Niech zdanie wejściowe do przetłumaczenia przy pomocy tego samego przykładu będzie następujące:

pl: „Od dawna mam samochód w garażu.”

Siatka odpowiedniości pomiędzy zdaniem wejściowym, a źródłowym wyglądałaby następująco:



Schemat S8 – siatka odpowiedniości

Byłaby z niej odczytana informacja, że słowo „samochód” odpowiada słowu „domu”, a kolejno słowo „garażu” odpowiada słowu „psa”. Po podstawieniu według schematu S7, zostanie opracowane tłumaczenie:

en: „I have had a garage in the car for a long time.”

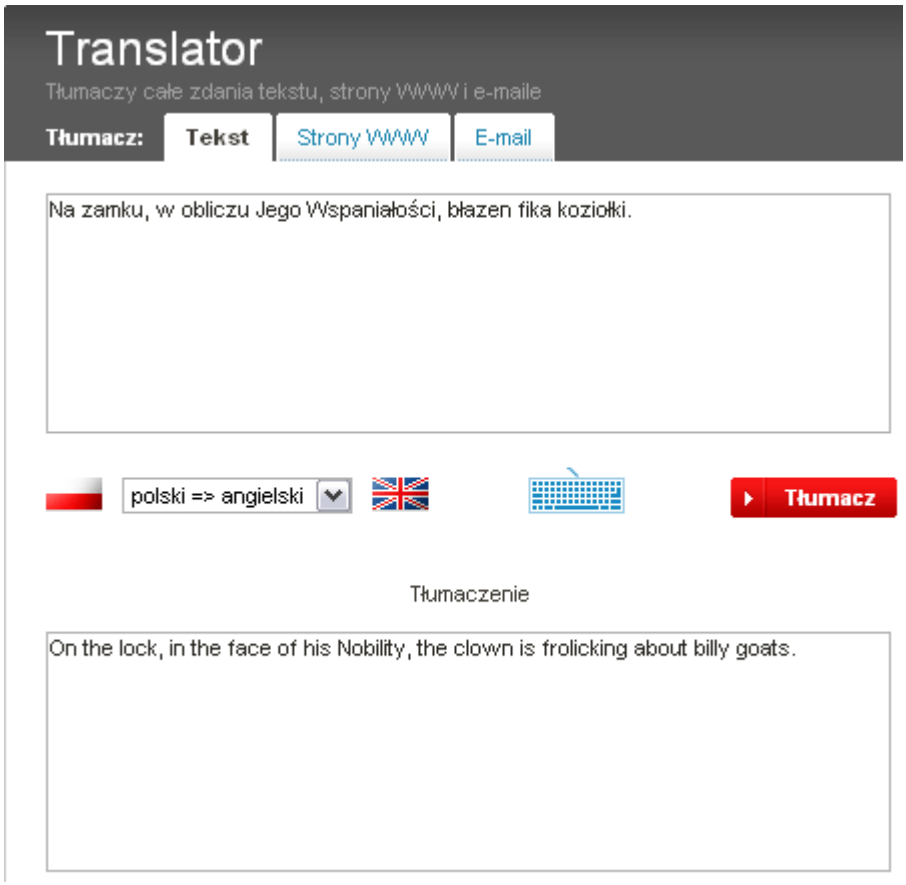
Nastąpiła niefortunna zamiana słów, spowodowana gorszym dopasowaniem przykładu. Algorytm, uwzględniając możliwość wystąpienia takich problemów, obniża dodatkowo ocenę tłumaczenia, w przypadku podstawiania więcej niż jednego słowa. Uwzględnia się tym samym możliwość podobnej zamiany kolejności słów w przetłumaczonym zdaniu.

ROZDZIAŁ 2
PODSTAWY TEORETYCZNE PRACY

2.1. TŁUMACZENIE PRZEZ ANALOGIE A TŁUMACZENIE PRZEZ TRANSFER

Tłumaczenie przez analogie można odnieść do tłumaczenia przez transfer. To drugie podejście jest następujące: program komputerowy otrzymuje na wejściu tekst, który ma przetłumaczyć. Wykonuje na nim liczne operacje (takie jak parsing, analiza semantyczna i wiele innych), które mają doprowadzić do tego, aby na wyjściu mogło pojawić się tłumaczenie tekstu wejściowego. Jakie operacje są wykonywane w środku, zależy od pomysłu projektantów systemu i od ich heurystycznego przeczucia, że te akurat operacje przyniosą oczekiwany skutek. Najczęściej jedynym wymogiem, jaki nakłada się na taki algorytm (oczywiście poza osiągnięciem dobrej jakości tłumaczenia) jest szybkość działania.

Siła algorytmu tłumaczącego zależy, co jest oczywiste, od skuteczności wewnętrznych mechanizmów obróbki tekstu. Bardzo trudno jednak jest przewidzieć z góry, jakie mechanizmy mają szansę powodzenia. Spowodowane jest to m.in. dużą nieregularnością i wieloznacznością języków naturalnych. Często wierzy się następującej heurystyce – aby tłumaczenie automatyczne się powiodło, należy dokonać jak najgłębszej i jak najdokładniejszej analizy tekstu do przetłumaczenia i zastosować wyrafinowane algorytmy samego transferu. Według tej zasady zostały zbudowane takie systemy tłumaczenia automatycznego jak Mikrokosmos Machine Translation Project (opisany w [Mikrokosmos, 1996]), Machine Translation Interlingua (opisany w [Interlingua, 1995-2006]) oraz system Translatica (opisany na [Translatica]). Niestety, tłumaczenia wykonywane przez te systemy dość często nie są wysokiej jakości:



Rysunek R1 – tłumaczenie wykonane przez system Translatica

Wobec zaistniałej sytuacji, zaczęto poszukiwania innych rozwiązań. Skoro nie zadziałały skomplikowane algorytmy tłumaczące, postawiono na prostotę rozwiązań. Takim właśnie, stosunkowo prostym rozwiązaniem, ma być idea tłumaczenia przez analogie.

Siłą tłumaczenia przez analogie ma być wykorzystanie w tłumaczeniu jedynie przykładu. Nie prowadzi się wnikliwej analizy zdania wejściowego. Pozwala to na uniknięcie popełnienia błędów, które zwykle popełnia się przy takiej analizie. Zamiast głęboko analizować, patrzy się jedynie na pojedyncze, drobne różnice między zdaniem wejściowym, a źródłowym. Następnie próbuje się (w ograniczonym stopniu) zniwelować te różnice.

Dużą zaletą tłumaczenia przez analogie jest również to, że system, który je realizuje, sam podaje ocenę wykonanego przez siebie tłumaczenia. Jest ona dla użytkownika miarą wiarygodności otrzymanego tłumaczenia. Co więcej, gdy w pamięci tłumaczeń nie zostanie odnaleziony odpowiedni przykład, tłumaczenie w ogóle nie zostanie dokonane. Jest to sytuacja znacznie bardziej korzystna, niż podanie tłumaczenia, które wprowadza w błąd. Dobry algorytm tłumaczenia przez analogie po-

winien więc tłumaczyć możliwie często, ale przede wszystkim tłumaczenie powinno być jak najbardziej poprawne, gdy już do niego dojdzie.

2.2. OPERACJE KONIECZNE DO PRZEPROWADZENIA SKUTECZNEGO TŁUMACZENIA PRZEZ ANALOGIE

Aby można było zastosować algorytm tłumaczący przez analogie, konieczne jest wcześniejsze wykonanie licznych operacji. Od ich powodzenia silnie zależy powodzenie samego tłumaczenia przez analogie.

2.2.1. Podział na zdania

Segmentacja jest pierwszą operacją, którą należy wykonać na tekście przed przystąpieniem do samego tłumaczenia. Zgodnie ze schematem S1, tekst do przetłumaczenia należy podzielić na zdania, by móc tłumaczyć go zdanie po zdaniu. (Można w tym miejscu wspomnieć, że segmentacja jest bardzo użyteczna również w przygotowywaniu pamięci tłumaczeń. Bardzo często pamięć tłumaczeń tworzy się w oparciu o pozyskane z różnych źródeł dokumenty, które nie są podzielone na zdania. Z uwagi na duże rozmiary takich dokumentów – pamięć tłumaczeń jest tym lepsza, im więcej zawiera przykładów – ich ręczny podział na zdania byłby zbyt pracochłonny).

Segmentacja nie jest, wbrew pozorom, operacją trywialną. Najprostszy algorytm dzieli tekst w miejscach kropek, wykrzykników i znaków zapytania. Rozważmy jednak następujące zdanie:

Sz. Pan mgr inż. Jan Kowalski, zamieszkały przy ul. Boh. Westerplatte (??), zgłosi się do W.K.U. w Poznaniu, celem m.in. przedstawienia zaświadczenia o odbyciu P.W. w m.st. Warszawie.

W oparciu o uproszczony algorytm segmentacji, zostanie dokonany podział:

Sz.
Pan mgr inż.
Jan Kowalski, zamieszkały przy ul.
Boh.
Westerplatte (?
?
), zgłosi się do W.
K.
U.
w Poznaniu, celem m.
in.
przedstawienia zaświadczenia o odbyciu P.
W.
w m.
st.
Warszawie.

Zamiast jednego zdania otrzymano aż 16!

W praktyce, do segmentacji stosuje się algorytm oparty o zbiór reguł segmentacji (taki jak opisany w pracy [Lipski, 2007]). Reguły te są najczęściej zapisane w formacie SRX (opisanym na [SRX]). Format ten dopuszcza dwa rodzaje reguł: reguły podziału, oraz reguły zabraniające podziału (wyjątki). Dla każdej reguły określa się, jaki tekst ma wystąpić przed miejscem podziału (lub braku podziału), a jaki po tym miejscu. Oto przykład reguły, która ustawia punkt podziału po znaku interpunkcyjnym, o ile zaraz za tym znakiem znajduje się biały znak:

<rule break="yes">	określenie, że jest to reguła podziału
<beforebreak>[\.\?!]</beforebreak>	wyrażenie regularne, określające to, co występuje przed miejscem podziału
<afterbreak>\s</afterbreak>	wyrażenie regularne, określające to, co występuje po miejscu podziału
</rule>	koniec reguły

Poniżej została z kolei zaprezentowana reguła, która zabrania dzielić zdania po skrócie *„ul.”*, jeśli zaraz za nim jest biały znak :

<rule break="no"> *określenie, że jest to reguła zabraniająca podziału*

<beforebreak>ul\.</beforebreak>

<afterbreak>\s</afterbreak>

</rule>

Dla zdefiniowanego zbioru reguł **reguły**, do podziału na zdania tekstu **tekst** stosuje się następujący algorytm podziału:

	Instrukcja	Komentarz
1	**S**:=0	Ustaw znacznik (kursor) na początek tekstu.
2	**L** := 1	Ustaw licznik reguł na 1
3	**found** := false	Znacznik odnalezienia pasującej reguły
4	if (**L** <= **reguły**.length) and (not **found**)	
5	**R** := **reguły**[**L**]	Pobranie bieżącej reguły
6	**L** := **L** + 1	
7	if (**tekst**[0..**S**-1] pasuje do **R**.beforebreak) and tekst[**S**..length(**tekst**)] pasuje do **R**.afterbreak)	Sprawdzenie, czy reguła ma w danym miejscu zastosowanie, polega na próbie dopasowania tekstu przed kursorem s do wzorca beforebreak reguły oraz tekstu za kursorem do wzorca afterbreak reguły.
8	**found** := true	
9	if <**R**.break> = yes	Jeśli reguła **R** jest regułą podziału
10	podziel zdanie w miejscu **S**	
11	Idź do 4	Jeśli reguła **R** nie jest regułą podziału (tylko wyjątkiem) nie następuje podział zdania. Podobnie jednak jak w przypadku reguły podziału, następuje zaprzestanie sprawdzania reguł i przejście do kolejnej pozycji w tekście (zmienna **found**, oznaczająca znalezienie pasującej reguły, ma wartość true)
12	Else	
13	if (**S** < length(**tekst**) - 1)	Jeśli nie jesteśmy jeszcze na końcu (ostatnią pozycją do sprawdzenia jest:

		S = length(**tekst**) -1)
	S := **S** + 1	
14	Idź do 2	
15	Else	
16	ZAKOŃCZ	

Algorytm podziału na zdania, wykorzystujący reguły SRX

Zgodnie z powyższym algorytmem, dla każdej pozycji w tekście znajduje się pierwszą pasującą regułę podziału lub braku podziału. To właśnie ta pierwsza znaleziona reguła znajduje zastosowanie dla danej pozycji w tekście. Z tego powodu ważna jest kolejność reguł. Na przykład, aby zapewnić, że tekst będzie dzielony na zdania po kropce, ale nie po skrócie *prof.*, należy najpierw podać wyjątek od reguły podziału, a dopiero potem właściwą regułę:

<rule break="no">
<beforebreak>prof.</beforebreak>
<afterbreak>\s</afterbreak>
</rule>

<rule break="yes">
<beforebreak>\.</beforebreak>
<afterbreak>\s</afterbreak>
</rule>

Powodzenie segmentacji zależy więc od przygotowania właściwych reguł podziału oraz wyjątków od nich. W praktyce, warto stosować różne zestawy reguł dla tekstów w różnych językach. Pozwala to uniknąć problemów z wzajemnym wykluczaniem się reguł podziału. Co więcej, w niektórych przypadkach uzasadnione jest nawet stosowanie kilku zestawów reguł w obrębie jednego języka. Właściwy zestaw wybiera się na podstawie klasy tekstu, który chcemy podzielić na zdania. Jest to uzasadnione na przykład dla tekstów prawniczych i medycznych. Skróty stosowane w tekstach prawniczych mogą być zupełnie różne od tych stosowanych w tekstach medycznych. Na przykład, jeśli ze względu na teksty prawnicze, określimy globalną regułę zabraniającą dzielenia zdania po skrócie *ust.* (od *ustęp*), w tekście medycznym nastąpi błąd podziału dla tekstu: *„Należy dbać o higienę ust. Pomaga w tym szczotkowanie zębów.”*

2.2.2. Dopasowywanie tekstów

Inną operacją istotną z punktu widzenia tłumaczenia przez analogie jest dopasowywanie (inaczej: urównoleglanie) tekstów [Lipski, 2007]. Pomaga ona w przygotowywaniu pamięci tłumaczeń. Najogólniej, dopasowywanie polega na przyporządkowaniu do siebie tych zdań z dwóch tekstów w dwóch różnych językach, które są swoimi tłumaczeniami.

Operację dopasowywania można bardziej formalnie zdefiniować następująco: Niech **S** będzie podzielonym na zdania tekstem w języku źródłowym, a **T** podzielonym na zdania tekstem w języku docelowym. Niech $\mathbf{s = (s_1,s_2,s_3,\ldots,s_n)}$, będzie ciągiem zdań tekstu **S**, a $\mathbf{t = (t_1,t_2,\ldots,t_m)}$ – ciągiem zdań tekstu **T**. **Urównolegleniem** tekstu **S** do tekstu **T** nazywamy ciąg **d** par $\mathbf{(c_s,c_t)}$, takich że:

- $\mathbf{c_s}$ **i** $\mathbf{c_t}$ są podciągami odpowiednio ciągów **s** i **t**
- przynajmniej jeden z ciągów $\mathbf{c_s}$ i $\mathbf{c_t}$ jest niepusty

Pary $\mathbf{(c_s,c_t)}$ nazywa się **dopasowaniami**. Dopasowaniem typu (i-j) nazywamy parę $\mathbf{(c_s,c_t)}$ taką, że długość ciągu $\mathbf{c_s}$ wynosi **i**, a długość ciągu $\mathbf{c_t}$ wynosi **j**.
Ciąg **d** musi spełniać następujące własności:

$$\bigcup_{(c_s,c_t)\in d}\ \bigcup_{z\in c_s} z = S$$

oraz

$$\bigcup_{(c_s,c_t)\in d}\ \bigcup_{z\in c_t} z = T$$

gdzie $\bigcup$ jest operacją konkatenacji (złączenia) tekstów.

Jak można odczytać z powyższej definicji, podczas operacji dopasowywania, żadne zdania nie mogą być pominięte lub zamienione w kolejności ani w tekście **S**, ani w **T**. Dzięki temu, dysponując tylko urównolegleniem tekstów (ciągiem **d**), można te teksty odtworzyć w takiej postaci, w jakiej były przed operacją dopasowania. Dobrym urównolegleniem nazywamy takie, dla którego w dopasowaniach $\mathbf{(c_s,c_t)}$ zdania z ciągu $\mathbf{c_t}$ są tłumaczeniem zdań z ciągu $\mathbf{c_s}$.

Najprostszym algorytmem dopasowania jest tzw. algorytm „jeden do jednego”. W tym algorytmie ciąg **d** ma postać:

$$d = \left(\left(\left(s_1\right),\left(t_1\right)\right),\left(\left(s_2\right),\left(t_2\right)\right),\left(\left(s_3\right),\left(t_3\right)\right),\ldots\right)$$

Taki algorytm nie jest wystarczający, gdyż nie uwzględnia rozbieżności, jakie zwykle występują pomiędzy tekstami. Wpływ na powstanie tych rozbieżności mają na przykład (za [Lipski, 2007]):

- różne liczby zdań użyte do wyrażenia tego samego w obu językach
- błędy spowodowane złą segmentacją tekstu
- pomyłkowe przeoczenie zdania przez tłumacza
- różnice w treści tekstów (zwykle na skutek aktualizacji tekstu **S** po przetłumaczeniu)

Szacuje się, że podobne problemy dotyczą około 10% wszystkich zdań w tekstach równoległych (za [Lipski, 2007]). Jednak gdy algorytm „jeden do jednego" napotka choćby jeden tego typu problem, dopasowania utworzone za nim będą błędne.

Z tego powodu opracowano algorytmy mniej wrażliwe na wyżej wymienione problemy. Są to między innymi:

Algorytm Gale'a Church'a

Jest to algorytm bazujący na długości zdań (patrz [Gale, Chuch, 1991]). Pod uwagę brane są wszystkie możliwe dopasowania. Dla każdego z nich obliczane jest prawdopodobieństwo dopasowania w oparciu o stosunek długości zdań z ciągu $\mathbf{c_s}$ do długości zdań z ciągu $\mathbf{c_t}$. Następnie ze wszystkich dopasowań wybiera się ciąg dopasowań o największym prawdopodobieństwie.

Wyróżnia się następujące zalety algorytmów dopasowania, bazujących na długości zdań (za [Lipski, 2007]):

- Niezależność od języków tekstów
- Niska złożoność czasowa
- Niska złożoność pamięciowa

Wadą jest jednak niska jakość urównolegleń dla trudnych tekstów, tj. takich, w których występuje wiele dopasowań innego typu niż 1 do 1.

Algorytm Moore'a

Algorytm ten działa analogicznie do algorytmu **Gale'a Church'a**, z tą jednak różnicą, że prawdopodobieństwo dopasowania obliczane jest nie w oparciu

o długość, ale o treść zdań. Algorytm próbuje dopasować te słowa ze zdań, które są swoimi tłumaczeniami. Aby taki algorytm mógł funkcjonować, konieczne jest wykorzystanie słowników. Prawdopodobieństwo dopasowania zdań jest wtedy obliczane dzięki metodom tłumaczenia statystycznego (patrz [Knight, 1999]). Tłumaczenie statystyczne jest techniką tłumaczenia automatycznego, bazującą (podobnie jak tłumaczenie przez analogie) na pamięci tłumaczeń. Jednak w odróżnieniu od tłumaczenia przez analogie, działa dzięki zbudowanym w oparciu o pamięć tłumaczeń statystycznym modelom tłumaczenia. Z takiego modelu można wydobyć informację, jakie jest prawdopodobieństwo, że dane zdanie źródłowe może być przetłumaczone na dane zdanie docelowe.

Podstawową zaletą algorytmów dopasowania bazujących na treści zdań jest wysoka jakość urównolegleń. Według danych zebranych przez autora pracy [Lipski, 2007], algorytm Moore'a pozwala uzyskać około 95% prawidłowych dopasowań typu (1-1).

Wadą algorytmu Moore'a (jak i wszystkich algorytmów bazujących na treści zdań) jest zależność od języków. Powoduje ona następującą niedogodność – gdy zachodzi potrzeba dopasowania tekstów w **n** różnych językach (dopasowując zawsze tylko 2 teksty na raz), należy na potrzeby algorytmu przygotować

$$L=\binom{n}{2}$$

słowników dwujęzycznych (dla każdego dwuelementowego zbioru języków). Opracowanie słownika w formie elektronicznej może być bardzo czasochłonną czynnością (projekty związane z opracowywaniem słowników elektronicznych mogą trwać nawet kilka lat).

Obliczanie podobieństwa dopasowania w algorytmie Moore'a jest znacznie bardziej skomplikowane, niż w algorytmie Gale'a-Church'a. Jednak dzięki optymalizacjom, zastosowanym w algorytmie Moore'a (opisanym w [Lipski, 2007]), nie występuje duża utrata szybkości działania algorytmu.

hunalign

Jednym z najbardziej rozpowszechnionych algorytmów urównoleglania jest hunalign (opisany na [hunalign]). Jest to darmowe oprogramowanie, opracowane na potrzeby większego projektu, mającego na celu zaimplementowanie tłumaczenia

statystycznego pomiędzy językiem węgierskim, a angielskim. Mimo to, nadaje się do urównoleglania tekstów w dowolnych językach.

Wejściem do algorytmu jest tekst podzielony na zdania oraz na tokeny. Wyjściem – ciąg par zdań w dwóch językach (pary te są nazywane *bisentences*).

Algorytm hunalign łączy cechy algorytmu Gale'a-Church'a oraz Moore'a. Kiedy jest dostępny słownik, do obliczania prawdopodobieństwa dopasowania wykorzystywana jest zarówno informacja o długości dopasowywanych zdań, jak i informacje słownikowe. W przypadku braku słownika, algorytm działa dwufazowo. Najpierw dopasowuje teksty, wykorzystując jedynie informację o długości zdań i na podstawie tego urównoleglenia buduje słownik. Następnie dokonuje dopasowania tekstów ponownie, tym razem korzystając nie tylko z informacji o długości zdań, ale także z nowo opracowanego słownika.

O popularności algorytmu zadecydowały następujące jego cechy: dostępność, wydajność i dokładność urównolegleń.

2.2.3. Wyszukiwanie przykładów w pamięci tłumaczeń

Do zrealizowania wyszukiwania odpowiednich par zdań, konieczne jest wykorzystanie funkcji, która szacuje podobieństwo między zdaniami. Opracowanie takiej funkcji okazuje się trudnym zagadnieniem.

Funkcja taka powinna otrzymać na wejściu dwa zdania w języku naturalnym, a zwrócić liczbę rzeczywistą z zakresu [0,1], która będzie miarą podobieństwa tych zdań. Wartość 1 oznacza zdania identyczne, natomiast 0 – zupełnie niepodobne.

Problem opracowania takiej funkcji może zostać rozwiązany na przykład dzięki indeksowi podobieństwa Jaccarda. Wtedy zdania, które mamy porównać, traktuje się jako zbiory słów. Indeks podobieństwa Jaccarda oblicza się ze wzoru:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

gdzie A to zbiór słów pierwszego zdania, a B – drugiego.

Naturalnie, przy wyznaczaniu sumy i różnicy zbiorów A i B, należy wiedzieć, jakie słowa uważa się za identyczne. Najprościej przyjąć, że słowa identyczne

to te, które nie różnią się między sobą nawet znakiem. Jednak rozszerzając to myślenie, można przyjmować za identyczne (lub chociaż podobne) słowa, które pochodzą z jednego leksemu (różne formy gramatyczne jednego słowa) lub nawet słowa będące jedynie synonimami.

Podejście Jaccarda nie uwzględnia kolejności słów w zdaniu. Aby je uwzględnić, opracowano zupełnie inną funkcję podobieństwa zdań. Jest ona rozszerzeniem odległości Levenshteina (opisana dokładnie w [Gintrowicz, 2007]). Odległość Levenshteina definiuje się dla dwóch napisów (ciągów znaków) jako „najmniejszą liczbę działań prostych, przeprowadzających jeden napis na drugi". Działaniami prostymi nazywamy:

- wstawienie nowego znaku do napisu
- usunięcie znaku z napisu
- zamianę znaku w napisie na inny znak

Miarę Levenstheina można w prosty sposób rozszerzyć na zdania. Wystarczy potraktować zdanie jako „napis", składający się z ciągu słów, pełniących rolę „znaków". Na przykład dla zdań:

1. *Dzisiaj jest ładna pogoda i słońce świeci jasno.*
2. *Dzisiaj jest brzydka pogoda i nie świeci słońce.*

policzmy działania proste, które przekształcają zdanie *1.* w *2.*:

1. zamiana słowa „ładna" na „brzydka"
2. wstawienie słowa „nie" za słowem „i"
3. usunięcie słowa „jasno"

Uogólniona odległość Levenshteina między tymi zdaniami wynosi więc *3*. Celem jest jednak obliczenie podobieństwa między zdaniami, a więc liczby rzeczywistej z przedziału [0,1]. Wobec tego, możemy na przykład zastosować formułę:

$\mathbf{M}_L = 1-(\mathbf{d}_L/\mathbf{s})$,

gdzie $\mathbf{M}_L$ to miara podobieństwa, $\mathbf{d}_L$ to odległość Levenshteina między zdaniami, a **s** to łączna liczba słów w obu zdaniach.

2.3. ROZPOZNAWANIE JEDNOSTEK NAZWANYCH (*NE*)

2.3.1. Jednostki nazwane i ich typy

Jednostka nazwana to spójna część tekstu, która zawiera określoną informację, taką jak nazwisko osoby lub nazwa organizacji ([NERT, 2008]). Wyróżnia się wiele typów jednostek nazwanych. Niektóre z nich przedstawiono poniżej:

Typ jednostki nazwanej	**Przykłady (w języku polskim)**
Osoba	Jan Kowalski Pani Anna Nowak prof. dr hab. inż. Piotr Nowakowski
Miejsce	Poznań Półwysep Helski Nowy Jork Przylądek Dobrej Nadziei
Organizacja	ART sp. z o. o. MZK SA
Data	12.03.2002 12 kwietnia 2002 roku 2002-12-03
Godzina	16:56 4:56 PM 4:56 PM (GMT)
Ilość pieniędzy	24 PLN 4.35 USD 87 złotych i 29 groszy

Tabela T1 – podstawowe typy jednostek nazwanych

W konkretnych zastosowaniach można identyfikować w tekście różne typy jednostek nazwanych. Podczas pracy nad klasą tekstów specjalistycznych, na przykład medycznych, wskazane jest wyszczególnianie równie wyspecjalizowanych jednostek nazwanych. Dla wspomnianych tekstów medycznych można zdefiniować klasy „nazwa leku”, „nazwa choroby” czy też „związek chemiczny”.

2.3.2. Przykład oznaczenia jednostek nazwanych w tekście

Niech dany będzie tekst w języku angielskim:

I met John Smith yesterday in London. He still works for the ACME Corporation

Zostaną w nim oznaczone jednostki nazwane, przy wykorzystaniu systemu LingPipe (opisany na [LingPipe]). Wynik zostanie podany w formacie XML.

Wynik oznaczenia

```
<output>
    <s i="0">
        I met
        <ENAMEX TYPE="PERSON">John Smith</ENAMEX>
        yesterday in
        <ENAMEX TYPE="LOCATION">London</ENAMEX>
        .
    </s>
    <s i="1">
        He still works for the
        <ENAMEX TYPE="ORGANIZATION">ACME Corporation</ENAMEX>
        .
    </s>
</output>
```

Komentarz

Jak można zaobserwować, system dokonał podziału tekstu na zdania. Zostały one umieszczone w oddzielnych węzłach <s>. Wewnątrz zdań zostały oznaczone jednostki nazwane. Każda z nich została umieszczona w węźle <ENAMEX>. Właściwość TYPE tego węzła określa typ oznaczonej jednostki nazwanej. W powyższym przykładzie występują następujące typy jednostek nazwanych:

PERSON – Osoba,
LOCATION – Miejsce,
ORGANIZATION – Organizacja.

System LingPipe umożliwia oznaczanie jednostek nazwanych związanych z konkretną tematyką. Do oznaczenia jednostek nazwanych w powyższym przykładzie użyto modułu „Breaking News". Służy on do oznaczania jednostek nazwanych w tekstach najświeższych wiadomości. Moduł bez trudu rozpoznał takie typy jednostek nazwanych jak Osoba, Miejsce i Organizacja.

2.3.3. Zastosowanie jednostek nazwanych w tłumaczeniu

Poprawne rozpoznawanie i tłumaczenie jednostek nazwanych odgrywa znaczącą rolę w procesie tłumaczenia automatycznego. Niepoprawne traktowanie jednostek nazwanych przez system tłumaczący najczęściej prowadzi do rażących błędów. Obrazuje to przykład, zaczerpnięty z pracy [NERT, 2008]:

Zdanie do przetłumaczenia (wejściowe):

„Podała rękę Pani Prezes Justynie Kowalskiej".

Poprawne tłumaczenie:

„(She) gave a hand to Mrs. Justyna Kowalska, Chairman*"*

Tłumaczenie wykonane przez system Translatica:

„A president extended a hand to Blacksmith's Justyna."

Komentarz

Dużym błędem, popełnionym przez system Translatica, jest przetłumaczenie nazwiska Kowalska na *Blacksmith*. (Co ciekawe, właśnie takie błędy najczęściej dyskwalifikują system tłumaczący w oczach potencjalnego użytkownika. System traci

przez nie swoją wiarygodność. Znaczenie systemu spada wtedy do poziomu zabawki, dostarczającej jedynie komicznych pomyłek). Nazwisko zostało przetłumaczone, gdyż znalazło się w słowniku, którym dysponuje system tłumaczący i żaden mechanizm nie zabronił słownikowego przetłumaczenia nazwiska Kowalska.

Jeszcze poważniejszym błędem było jednak błędne zinterpretowanie zdania wejściowego. Jak można odczytać z tłumaczenia, wykonanego przez Translatikę, słowo Prezes zostało uznane za podmiot zdania, tak jakby zdanie wejściowe brzmiało:

„Prezes Podała rękę Pani Justynie Kowalskiej”

System tłumaczący nie popełniłby tych błędów, gdyby wiedział, że wyrażenie „Pani Prezes Justynie Kowalskiej” jest pojedynczą jednostką nazwaną typu Osoba.

Jak wynika z cytowanego w pracy [NERT, 2008] raportu D. Vilar’a, dotyczącego częstości błędów popełnianych przez systemy tłumaczenia automatycznego, błędy polegające na niewłaściwym rozpoznaniu jednostek nazwanych stanowią około 10% wszystkich błędów popełnianych przez te systemy. Jednak według analiz przeprowadzonych przez autora pracy [NERT, 2008] błędy takie są nawet częstsze. Występują szczególnie często podczas analizy języków naturalnych, w których szyk zdania może się zmieniać (jest tak na przykład w językach grupy słowiańskiej). Właśnie zmienny szyk zdania był przyczyną pomyłki systemu tłumaczącego podczas tłumaczenia w ramach przytoczonego wyżej przykładu.

2.3.4. Mechanizmy rozpoznawania jednostek nazwanych

Uznaje się (za [NERT, 2008]), że historia rozpoznawania jednostek nazwanych rozpoczęła się w 1996 roku, na szóstej Message Understanding Conference (MUC-6). Pierwsze podejścia do tego zagadnienia polegały na wykorzystaniu specjalnie do tego celu zaprojektowanych reguł (tzw. Hand Crafted Rules, lub HC rulet). Reguły te były konkretnymi wytycznymi, określającymi w jaki sposób mają być wyszukiwane jednostki nazwane w tekście. Wszystkie reguły musiały być opracowane przez człowieka.

W ramach rozwoju idei rozpoznawania jednostek nazwanych, zdecydowano się wykorzystać uczenie maszynowe. Reguły oznaczania jednostek nazwanych były wnioskowane przez system.

Uczenie maszynowe w systemach oznaczających jednostki nazwane dzieli się na następujące kategorie [NERT, 2008]:

Uczenie nadzorowane

Mechanizmy oznaczania jednostek nazwanych są wnioskowane ze znacznej wielkości korpusów językowych, a następnie sprawdzane przez człowieka.

Uczenie nienadzorowane

Proces pozyskiwania reguł oznaczania jednostek nazwanych nie jest kontrolowany przez człowieka. Zamiast tego, wykorzystuje się istniejące sieci semantyczne leksemów, takie jak WordNet.

Uczenie półnadzorowane

Proces uczenia maszyny zakłada niewielką ingerencję człowieka. Może się ona przejawiać jako dostarczenie początkowych reguł oznaczania jednostek nazwanych, na podstawie których pozyskiwane są następne.

ROZDZIAŁ 3
PRZYKŁADY SYSTEMÓW TŁUMACZĄCYCH PRZEZ ANALOGIE

3.1. ALGORYTM OPRACOWANY PRZEZ MAKOTO NAGAO

Jest to dość wczesny, bo pochodzący z 1984 roku, opis algorytmu tłumaczącego przez analogie, głównie między językiem japońskim i angielskim (patrz: [Nagao, 1984]). Uważa się nawet, że były to pierwsze na świecie przymiarki do tego sposobu tłumaczenia, a Makoto Nagao jest ojcem idei tłumaczenia przez analogie.

3.1.1. Idea algorytmu

Od Nagao pochodzi uzasadnienie opierania tłumaczenia na przykładach (mechanizm ten ma być naturalnym sposobem tłumaczenia, wykorzystywanym przez człowieka). Wobec tego, w algorytmie Nagao widać dążenie do obdarzenia maszyny wiedzą oraz mechanizmami, jakie są wykorzystywane przez człowieka w tłumaczeniu.

Sam algorytm wpisuje się w ramy ogólnego, sześciopunktowego algorytmu tłumaczenia przez analogie, przedstawionego w rozdziale 1.1. Jedyną różnicą jest brak podawania oceny tłumaczenia. Cechą charakterystyczną algorytmu Nagao jest natomiast fakt, że podczas wykonywania kroków 2 i 3 wykorzystywana jest wiedza na temat tego, kiedy i jakie różnice między zdaniami można przenosić.

Wiedza ta pochodzi ze specjalnie skonstruowanej pamięci tłumaczeń. Konstrukcja ta przebiega, w myśl przyświecającej idei, podobnie jak u człowieka, uczącego się nowego języka. Swoiste uczenie maszyny polega na nauczeniu jej najpierw prostego zdania (to znaczy umieszczenia w pamięci tłumaczeń prostego przykładu). Następnie, zamienia się jeden wyraz w zdaniu źródłowym i uczy się maszynę

tłumaczyć tak powstałe zdanie (tj. dodaje się do pamięci tłumaczeń nowy przykład). Takie postępowanie powtarza się wielokrotnie, dla dużej ilości zdań.

Dzięki tak skonstruowanej pamięci tłumaczeń, możliwe jest wywnioskowanie reguł podstawiania fraz w zdaniach. Ogólny mechanizm takiego wnioskowania (zaczerpnięty wprost z pracy Makoto Nagao), zamieszczono poniżej:

Pamięć tłumaczeń:

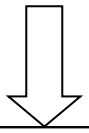
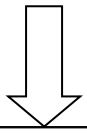
	Język angielski	Język japoński
Przykład 1:	α X β	α' X' β'
		
Przykład 2:	α Y β	α' Y' β'

Tabela T2 – fragment pamięci tłumaczeń w systemie M. Nagao

Literami α,β,α' i β' oznaczono kontekst, w jakim występują frazy X,Y,X' i Y'.

Wnioski podczas budowania reguł:

- Kontekst α – β w języku angielskim odpowiada kontekstowi α' – β' w języku japońskim.
- Frazę X należy tłumaczyć na X'
- Frazę Y należy tłumaczyć na Y'

Reguły te pozwalają na podstawianie całych fraz, a nie tylko pojedynczych słów, jak w mechanizmach zaprezentowanych w rozdziale 1.3.

3.1.2. Działanie algorytmu

Aby zobrazować sposób wnioskowania reguł podmiany na podstawie pamięci tłumaczeń, zostanie przedstawiony przykład. Niech językiem źródłowym będzie język polski, a docelowym – angielski. Niech w pamięci tłumaczeń znajdują się następujące dwa przykłady:

Przykład 1:

pl: „Zostało powiedziane, że tłumaczenie przez analogie ma szansę dokonać rewolucji w nauce”.

en: „It has been said that example based machine translation has a chance of revolutionizing science”.

Przykład 2:

pl: „Zostało powiedziane, że rodzicielskie piętno genomowe ma szansę dokonać rewolucji w nauce”.

en: „It has been said that genomic imprinting has a chance of revolutionizing science”.

Podczas wnioskowania reguł podstawiania fraz, odpowiednie parametry przyjmują następujące wartości:

Parametr	**Znaczenie**	**Wartość**
α	Lewy kontekst w zdaniach źródłowych	*Zostało powiedziane, że*
β	Prawy kontekst w zdaniach źródłowych	*ma szansę dokonać rewolucji w nauce*
α’	Lewy kontekst w zdaniach docelowych	*It has been said that*
β’	Prawy kontekst w zdaniach docelowych	*has a chance of revolutionizing science*
X	Fraza do podstawienia w pierwszym przykładzie	*tłumaczenie przez analogie*
X’	Przetłumaczona fraza w pierwszym przykładzie	*example based machine translation*
Y	Fraza do podstawienia w drugim przykładzie	*rodzicielskie piętno genomowi*
Y’	Przetłumaczona fraza w drugim przykładzie	*genomic imprinting*

Tabela T3 – parametry wykorzystywane podczas działania algorytmu M. Nagao

Zgodnie z wcześniej opisanym mechanizmem, algorytm w tej sytuacji wnioskuje, że fraza *„tłumaczenie przez analogie”* powinna być tłumaczona jako *„example based machine translation”*, a fraza *„rodzicielskie piętno genomowe”* jako *„genomic imprinting”*. Jest to wnioskowanie całkowicie poprawne, dzięki któremu można uzyskać wysokiej jakości tłumaczenie. Co więcej, nawet człowiek mógłby popełnić błąd w tłumaczeniu frazy *„rodzicielskie piętno genomowe”*, jeśli nie znałby terminologii biologicznej.

3.1.3. Dyskusja nad algorytmem

Opisane mechanizmy wydają się być wystarczające do przeprowadzania bardzo skutecznego tłumaczenia. Niestety, nie zawsze jest to możliwe.

Pierwszym problemem w stosowaniu tego algorytmu jest przygotowanie pamięci tłumaczeń. Tłumaczenia muszą być dokładne, pamięć musi zawierać wiele podobnych do siebie przykładów i, przede wszystkim, ilość przykładów musi być bardzo duża. Trudności, jakie napotyka się w praktyce przy tworzeniu pamięci tłumaczeń, uniemożliwiają przygotowanie jej w taki sposób.

Poważniejszy problem leży jednak w logice algorytmu. Tworzenie reguł tłumaczenia fraz na podstawie pamięci tłumaczeń może prowadzić do uzyskania błędnych reguł. Rozważmy następującą sytuację:

Próba wnioskowania reguł w oparciu o następujące przykłady:

Przykład 1:

pl: „W teatrze wystawiono wspaniałą sztukę”.

en: „A great play has been performed at the theatre”.

Przykład 2:

pl: „W teatrze ogromna publiczność ogląda sztukę”.

en: „A great audience is watching the play at the theatre”.

Odpowiednie parametry przyjmą wtedy wartości:

Parametr	**Wartość**
α	*W teatrze*
β	*sztukę*
α’	*A great*
β’	*at the theatre*
X	*wystawiono wspaniałą*
X’	*play has been performed*
Y	*ogromna publiczność ogląda*
Y’	*audience is watching the play*

Tabela T4 – wartości parametrów podczas działania algorytmu M. Nagao

Otrzymujemy więc błędne reguły tłumaczenia:

„wystawiono wspaniałą” na *„play has been performed”*

oraz

„ogromna publiczność ogląda” na *„audience is watching the play”*.

Błąd był spowodowany innym szykiem zdań w języku źródłowym i docelowym.

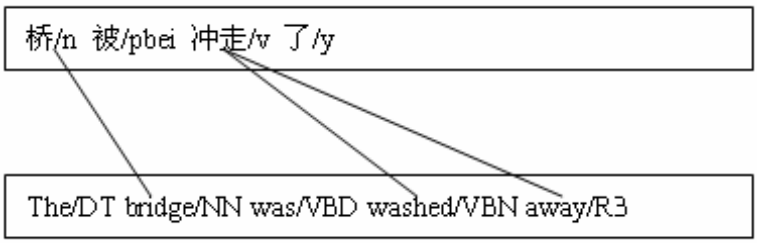
Wymienione słabe strony algorytmu Makoto Nagao skłaniają do poszukiwania innych algorytmów tłumaczenia przez analogie.

3.2. SYSTEM CHIŃSKIEJ AKADEMII NAUK

Drugim spośród systemów, które zostaną tu opisane, jest system tłumaczenia przez analogie, opracowany przez Chińską Akademię Nauk. Językiem źródłowym w tym systemie jest język chiński, a docelowym – angielski. System korzysta z 220000 przykładów. Jest on dokładnie opisany w artykule [CAS, 2004].

3.2.1. Idea systemu

Algorytm odbiega nieco od modelu tłumaczenia przez analogie, opisanego w pierwszym rozdziale niniejszej pracy. Najbardziej zasadniczą różnicą jest to, że korzysta on z pamięci tłumaczeń dopasowanej na poziomie słów. Oznacza to, że zawiera ona nie tylko przykładowe tłumaczenia zdań, ale także informację o tym, które słowa w danych zdaniach odpowiadają sobie. Ponadto, dzięki mechanizmowi oznaczania części mowy (*POS tagging*), przechowuje informację, jaką częścią mowy jest każdy wyraz w zdaniu. Oto, jak wygląda przykład z pamięci tłumaczeń tego systemu (zaczerpnięty z artykułu [CAS, 2004]):



Schemat S9 – siatka odpowiedniości w systemie Chińskiej Akademii Nauk

Strzałki obrazują dopasowanie słów (należy zwrócić uwagę, że jednemu słowu chińskiemu może odpowiadać wiele słów angielskich). Za każdym wyrazem, po znaku „/”

(ukośniku) znajduje się symbol części mowy, która odpowiada temu wyrazowi (np. NN oznacza rzeczownik).

Sam algorytm tłumaczenia zdania realizuje ogólny algorytm, opisany w rozdziale 1.1. Specyfika algorytmu opracowanego przez Chińską Akademię Nauk jest następująca:

- Do wyszukiwania zdań w pamięci tłumaczeń używana jest funkcja porównująca fragmenty dwóch zdań. Miarą podobieństwa jest długość najdłuższego dopasowanego fragmentu.
- Podstawianie słów następuje podobnie jak w przykładach z rozdziału 1.3. Wykorzystuje się jednak informację o częściach mowy podstawianych słów. Dzięki niej wykonuje się tylko takie podstawienia, które mają mniejszą szansę spowodowania powstania błędów gramatycznych w wyniku tłumaczenia.
- Całość kończy przetwarzanie końcowe, które dokonuje kosmetycznych poprawek w wyniku tłumaczenia.
- System nie podaje oceny tłumaczenia

3.2.2. Dyskusja nad systemem

Szczegółowe testy systemu wykazały, że w większości przypadków wyniki są niezadowalające. Wyszczególniono następujące słabe punkty systemu:

- Podział tekstów na zdania jest niskiej jakości (często powstają zbitki tekstu).
- Dopasowywanie na poziomie słów jest często przeprowadzane błędnie.
- Zdania w przykładach często różnią się występowaniem niektórych słów.
- Tłumaczenie krótkich fragmentów jest niskiej jakości.

Podsumowując, do zrealizowania dobrego tłumaczenia przez analogie, konieczne jest wypracowanie innego podejścia.

ROZDZIAŁ 4
SYSTEM HETMAN

4.1. KRÓTKA CHARAKTERYSTYKA SYSTEMU I JEGO ZASTOSOWANIA

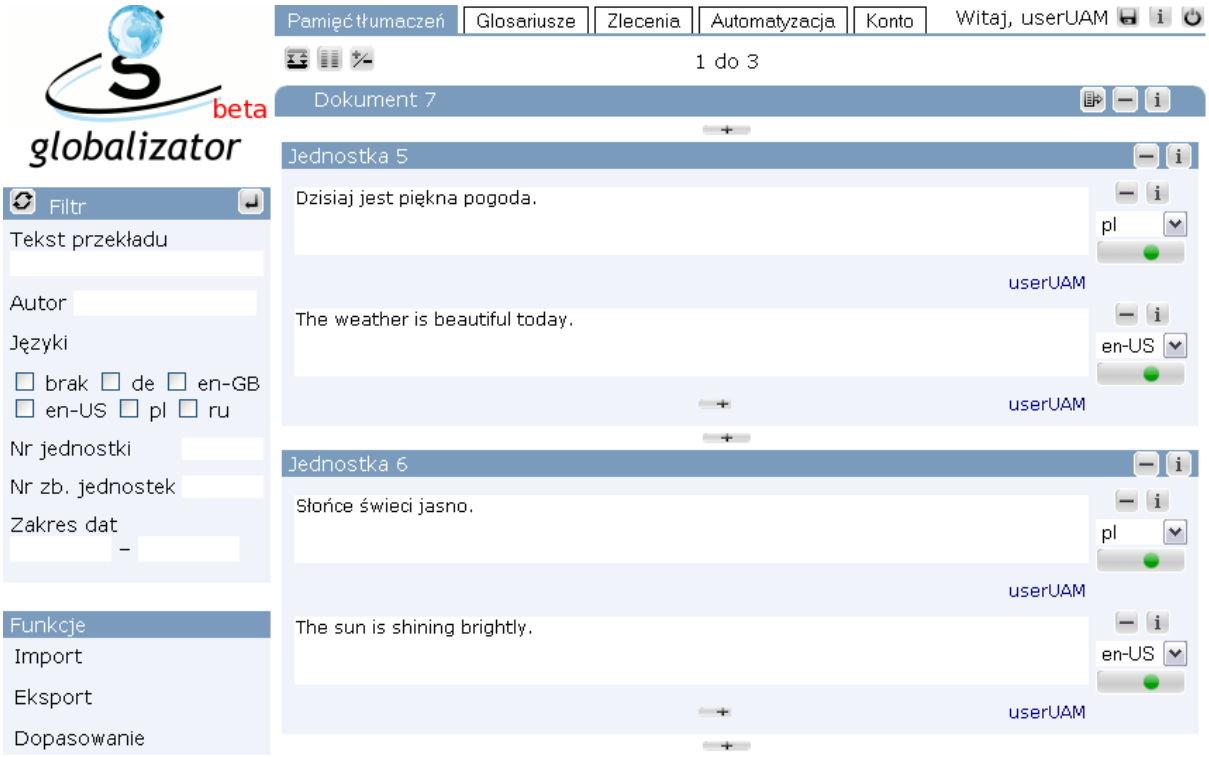
4.1.1. Informacje ogólne o systemie

System heTMan powstał i jest nadal rozwijany jako komercyjny produkt firmy PolEng. Prace nad systemem trwają już około trzech lat. Jest on rozprowadzany razem z systemem Translatica Server - serwerem tłumaczenia automatycznego (opisanym na [Translatica Server]). Poza tym, heTMan stanowi interfejs użytkownika w oferowanej przez firmę PolEng usłudze Globalizacji (opisanej na [Globalizacja]). Jest to usługa polegająca na tłumaczeniu witryn internetowych przy udziale tłumaczy, wspomaganych technikami tłumaczenia automatycznego.

System heTMan realizuje szereg funkcjonalności związanych z automatycznym przetwarzaniem tekstu, wspomaganiem tłumaczenia oraz tłumaczeniem automatycznym. Posiada budowę modularną, dzięki czemu można łatwo rozwijać każdy moduł z osobna i włączać je do systemu głównego w zależności od potrzeb. W rozdziale 4.1 zostaną zaprezentowane główne moduły systemu heTMan.

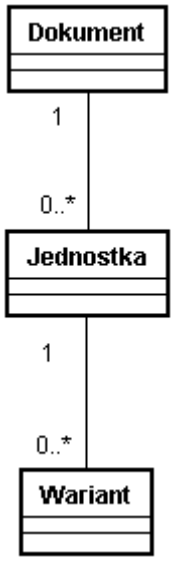
4.1.2. Zarządzanie pamięcią tłumaczeń

Moduł zarządzania pamięcią tłumaczeń to najważniejszy, główny moduł systemu. Jest on wymagany do poprawnego działania heTMana. Służy do zarządzania przykładami w pamięci tłumaczeń. Umożliwia dodawanie, modyfikację oraz usuwanie przykładów.



Rysunek R2 – zarządzanie pamięcią tłumaczeń w systemie heTMan w wersji Globalizator

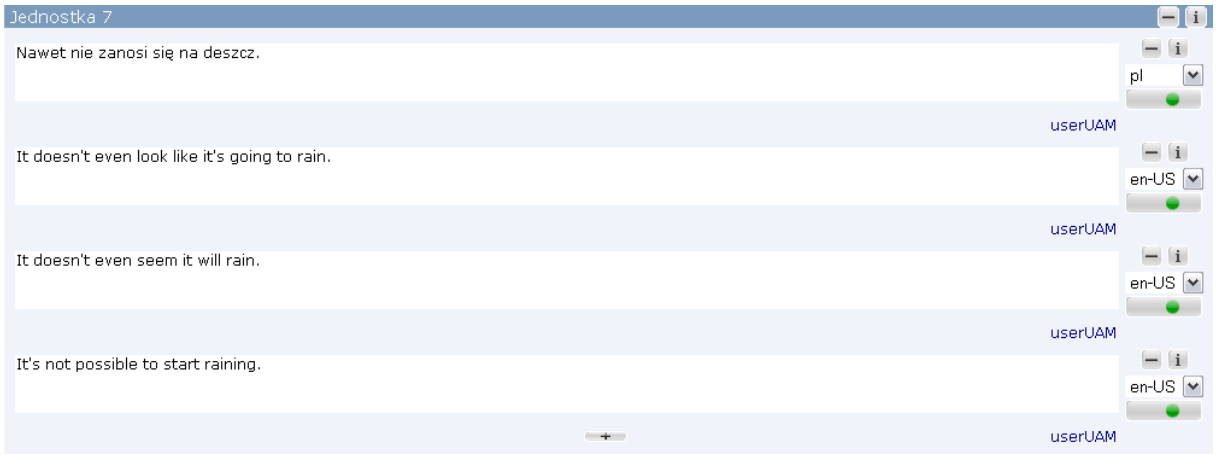
Organizacja pamięci tłumaczeń w systemie heTMan opiera się na standardzie wymiany pamięci tłumaczeń TMX (Translation Memory Exchange, opisany w [TMX]). Ogólny schemat tej organizacji jest następujący:



Rysunek R3 – schemat organizacji pamięci tłumaczeń w heTManie

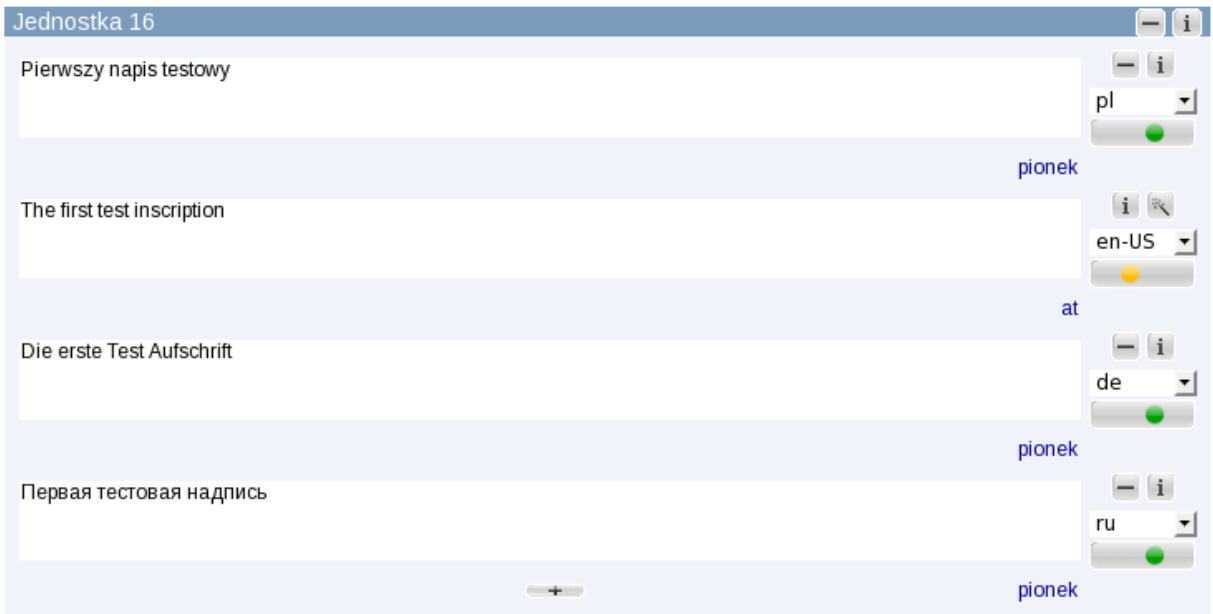
Dokument jest największym obiektem, przechowywanym w pamięci tłumaczeń. Reprezentuje zapisany w dowolnym formacie dokument, zawierający tekst, który stał się źródłem jednostek. Ogólnie – jest to zbiór jednostek.

Jednostka tłumaczenia jest rozszerzeniem pojęcia przykładu w pamięci tłumaczeń. Reprezentuje ona zdanie lub część tekstu wraz ze wszystkimi możliwymi tłumaczeniami. Może to być na przykład zdanie polskie wraz z kilkoma wariantami tłumaczeń angielskich:



Rysunek R4 – jednostka tłumaczenia: jedno zdanie polskie – trzy warianty tłumaczenia w języku angielskim

Jednostkę może jednak stanowić również na przykład zdanie polskie przetłumaczone na różne języki:



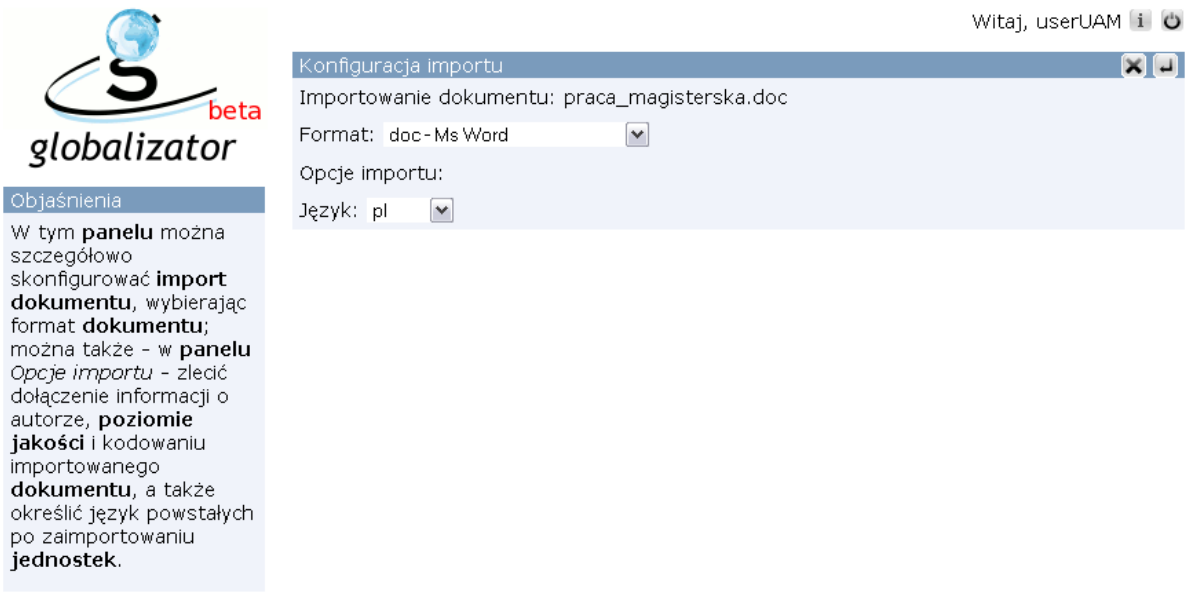
Rysunek R5 – jednostka tłumaczenia: zdanie polskie przetłumaczone na trzy języki

Wariant tłumaczenia to pojedyncze zdanie lub część tekstu w określonym języku, wchodzące w skład jednostki. Jednostka z rysunku R5 zawiera 4 warianty.

4.1.3. Import i eksport dokumentów

System heTMan umożliwia importowanie dokumentów, zapisanych w różnych formatach, do zarządzanej przez siebie pamięci tłumaczeń. Proces importu dokumentu polega na przeniesieniu jego treści tekstowej do pamięci tłumaczeń, zarządzanej przez system heTMan. Podczas importu dokumentu, wykonywane są następujące czynności:

1. Pozyskanie treści dokumentu w postaci czystego tekstu (tj. bez znaków formatowania i znaczników)
2. Podział tekstu na zdania
3. Utworzenie nowego dokumentu w pamięci tłumaczeń
4. Utworzenie po jednej jednostce dla każdego zdania, powstałego w wyniku podziału tekstu dokumentu na zdania
5. Dla każdej z utworzonych jednostek utworzenie jednego wariantu tłumaczenia, zawierającego treść zdania



Rysunek R6 – importowanie dokumentu

Obecnie obsługiwane formaty dokumentów wejściowych to:

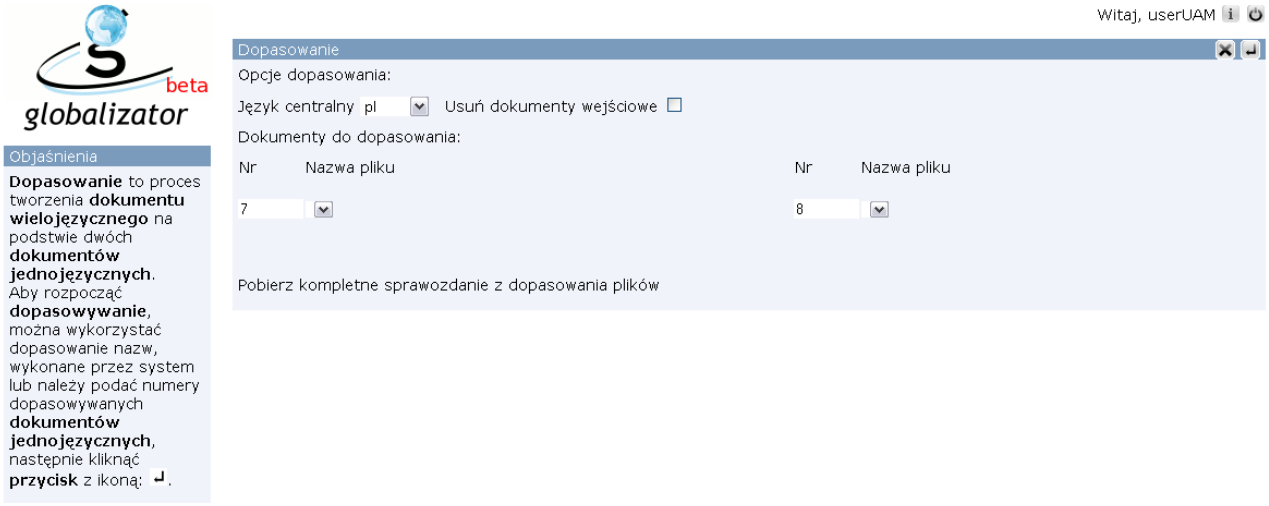
- doc (dokument Microsoft Word)

- tmx (plik pamięci pamięci tłumaczeń, w standardzie opisanym w [TMX])
- ppt (plik prezentacji programu Microsoft PowerPoint)
- xls (arkusz kalkulacyjny programu Microsoft Excel)
- html (dokument znaczników)
- zip (archiwum ZIP plików. Spakowane pliki mogą być w dowolnych z wymienionych wyżej formatów)

System heTMan pozwala również na eksport jednostek, przechowywanych w swojej pamięci tłumaczeń. Wyeksportowane jednostki zapisuje w formacie tmx lub txt (zwykły tekst).

4.1.4. Dopasowywanie dokumentów

System heTMan pozwala na dopasowywanie dokumentów.



Rysunek R7 – dopasowywanie dokumentów w hetmanie

Aby rozpocząć dopasowywanie, należy podać numery dokumentów do dopasowania. Opcja „język centralny”, określa język pierwszego dokumentu z pary dokumentów do dopasowania. Pomaga ona w przygotowaniu listy dokumentów do dopasowania. W ramach dopasowywania, tworzony jest nowy dokument z jednostkami zawierającymi warianty z dopasowywanych dokumentów. Opcjonalnie możliwe jest usunięcie dopasowywanych dokumentów, a pozostawienie jedynie dokumentu, powstałego w wyniku dopasowania.

4.1.5. Globalizacja

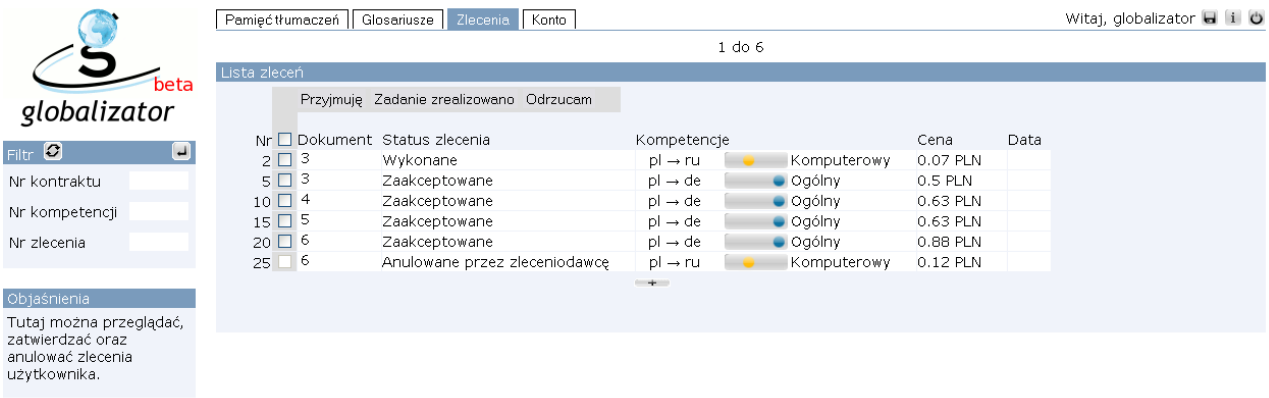
Globalizacja to usługa tłumaczenia portali internetowych. Klientem tej usługi jest administrator portalu internetowego, który posiada już portal w jednym języku i chce opublikować go w innych językach. Dzięki globalizacji, klient może zlecić przetłumaczenie treści swojego portalu profesjonalnym tłumaczom. Po zakończeniu tłumaczenia, przetłumaczone treści zostaną automatycznie umieszczone w portalu klienta.

Usługa tłumaczenia dotyczy również wszystkich zmian, jakie pojawiają się w portalu. Każda nowa treść, pojawiająca się w portalu, jest automatycznie przekazywana do tłumaczy, by po przetłumaczeniu, trafić z powrotem do portalu.

Dostarczycielem usługi globalizacji jest firma POLENG Sp. z o.o. Za realizację usługi Globalizacji odpowiada aplikacja Synchronizator oraz system heTMan.

Synchronizator odpowiada za pozyskiwanie treści do przetłumaczenia z portalu klienta, oraz za umieszczanie przetłumaczonych tekstów z powrotem w tym portalu. Samo tłumaczenie odbywa się za pośrednictwem systemu heTMan w wersji Globalizator.

Każdy artykuł z portalu klienta, który ma zostać przetłumaczony, jest importowany do systemu heTMan jako osobny dokument. Następnie, dla tak zaimportowanego dokumentu tworzone jest osobne zlecenie tłumaczenia. System heTMan umożliwia zarządzanie zleceniami tłumaczenia:



Rysunek R8 – zarządzanie zleceniami tłumaczenia w hetmanie

System heTMan dba o to, by odpowiednie zlecenia tłumaczenia zostały przydzielone do właściwych tłumaczy. Na przykład zleceniem tłumaczenia nr 5,

przedstawionym na rysunku R8 musiałby zająć się germanista, a zleceniem nr 2 – rusycysta.

Realizacja zleceń tłumaczenia odbywa się poprzez edytowanie odpowiednich wariantów przez tłumacza z wykorzystaniem interfejsu zaprezentowanego w rozdziale 4.1.2.

Cykl życia zlecenia tłumaczenia może być skomplikowany. Poniższa tabela przedstawia wszystkie możliwe statusy zleceń tłumaczenia:

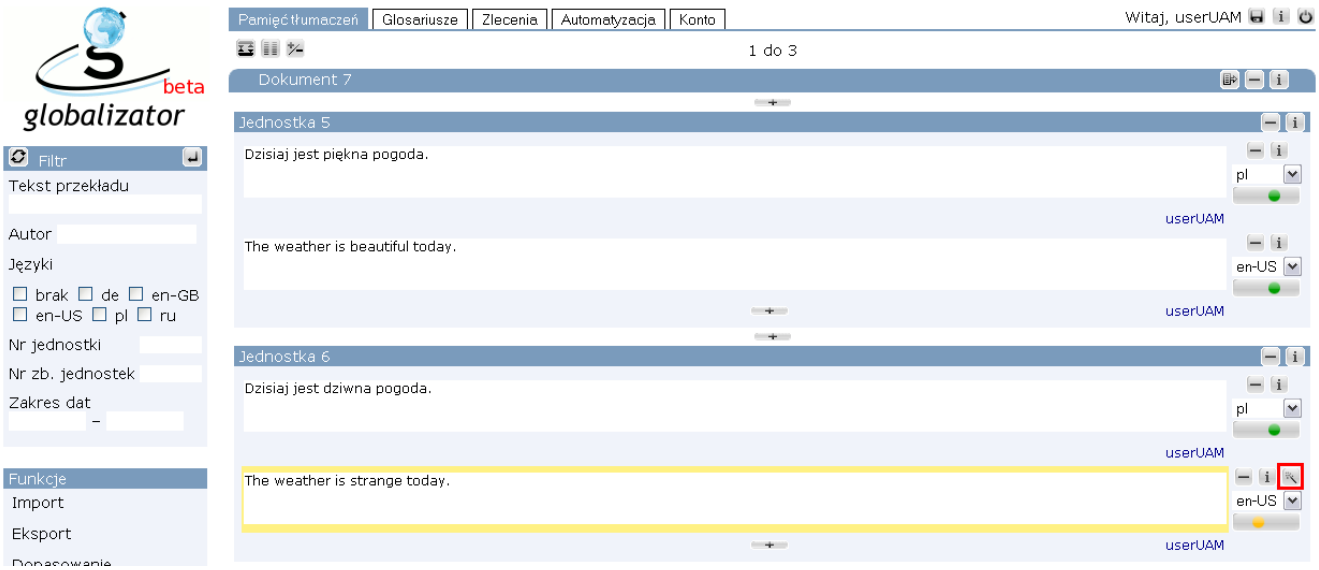
Nazwa statusu	**Komentarz**
Utworzone	Zlecenie nowoutworzone
Znormalizowane	Zlecenie przetworzone przez system heTMan, do formatu akceptowanego przez ten system
Wycenione	Zlecenie wycenione automatycznie przez system heTMan (na podstawie cenników pracy tłumaczy)
Zaakceptowane	Zlecenie zaakceptowane przez klienta (klient zaakceptował podaną przez system cenę)
Anulowane	Zlecenie anulowane przez klienta (klient nie zaakceptował ceny podanej przez system, bądź z jakichkolwiek innych przyczyn nie zgadza się na realizację zadania. Opcja anulowania zlecenia jest przydatna, gdyż zlecenia są najczęściej generowane automatycznie i mogą zostać wygenerowane błędnie).
Przydzielone	Zlecenie zaakceptowane przez klienta, które zostało przydzielone odpowiedniemu tłumaczowi
Realizowane	Zlecenie przyjęte przez tłumacza do realizacji i realizowane
Odrzucone	Zlecenie nieprzyjęte przez tłumacza do realizacji
Ukończone	Zlecenie ukończone przez tłumacza

Tabela T5 – statusy zleceń tłumaczenia w systemie Globalizator

Więcej informacji o usłudze globalizacji można znaleźć na stronie [Globalizacja].

4.1.6. Tłumaczenie przez analogie w systemie heTMan

System heTMan umożliwia również wykonanie tłumaczenia przez analogie. Służy do tego przycisk oznaczony różdżką.



Rysunek R9 – tłumaczenie przez analogie w systemie heTMan

Tłumaczenie jest w istocie wykonywane tak zwaną metodą hybrydową. Najpierw próbuje się wykonać tłumaczenie przez analogie, na bazie aktualnej pamięci tłumaczeń, przechowywanej przez system. Jeśli tłumaczenie to nie powiedzie się lub jego ocena będzie za niska, zostanie wykonane tłumaczenie metodą transferu, za pomocą systemu Translatica.

4.2. ZAGADNIENIA IMPLEMENTACYJNE

4.2.1. Struktura systemu heTMan

System heTMan realizuje architekturę klient-serwer. Serwer zajmuje się realizowaniem żądań klienta, które zgłaszane są za pomocą interfejsu WWW. System jest w całości napisany w języku Java. Serwer korzysta z silnika Apache Tomcat, natomiast interfejs WWW jest zrealizowany przy użyciu technologii JSP. Ta ostatnia technologia pozwala na używanie języka Java do generowania stron w języku HTML, czytelnych dla najpopularniejszych przeglądarek internetowych (Internet Explorer, Mozilla Firefox).

Jak już zostało wspomniane, heTMan ma budowę modularną. W obecnej wersji systemu istnieje kilkadziesiąt modułów. Czasem są one rozwijane w ramach oddzielnych projektów i dopiero później włączane do heTMana. Najważniejsze moduły heTMana zostały wymienione w tabeli T6:

Lp.	Nazwa	Zastosowanie	Okoliczności powstania
1	Core	Główny moduł systemu – podstawowy interfejs do zarządzania pamięcią tłumaczeń	Rozwijany od początku prac nad systemem
2	Splitter	Dzielenie tekstu na zdania, z wykorzystaniem reguł w formacie SRX	Realizowany w ramach pracy magisterskiej Jarosława Lipskiego [Lipski, 2007]
3	Aligner	Dopasowanie tekstów	Realizowany w ramach pracy magisterskiej Jarosława Lipskiego [Lipski, 2007]
4	I/O	Import i eksport dokumentów	Napisany w ramach prac nad rozwojem systemu
5	ANTRA	Tłumaczenie przez analogie	Realizowany w ramach pracy magisterskiej Jacka Gintrowicza [Gintrowicz, 2007]
6	WS	Interfejs dostępu do systemu przez usługi sieciowe	Napisany w ramach prac nad rozwojem systemu
7	NeLex	Nowy system tłumaczenia przez analogie	Realizowany w ramach niniejszej pracy magisterskiej
8	Wordtools	Przechowywanie słowników oraz reguł rozpoznawania jednostek nazwanych	Realizowany w ramach niniejszej pracy magisterskiej

Tabela T6 – wykaz modułów systemu heTMan

4.2.2. Tłumaczenie przez analogie w systemie heTMan

Za tłumaczenie przez analogie w systemie heTMan odpowiadał dotychczas jeden z jego modułów – system ANTRA. Był to system zrealizowany przez Jacka Gintrowicza, w ramach jego pracy magisterskiej, obronionej w 2007 roku [Gintrowicz, 2007].

System ten podporządkowuje się wszystkim ogólnym wytycznym, przedstawionym w rozdziale 1.1. Posiada więc identyczną specyfikację wejścia i wyjścia, co nowoopracowywany algorytm, który zostanie przybliżony w rozdziale 5.

Architektura systemu ANTRA jest trójwarstwowa. Za realizację zadań mieszczących się w każdej warstwie odpowiadają składowe systemu, nazwane przez jego autora pakietami. Wyróżnia się następujące pakiety:

- Pakiet transferu wzorcowego
- Pakiet transferu słownikowego
- Pakiet oceny jakości tłumaczenia

Podczas pracy nad tłumaczeniem zdania, pakiety te wykorzystywane są kolejno, jeden po drugim.

Pakiet transferu wzorcowego

Odpowiada on za tłumaczenie przez analogie z wykorzystaniem wyrażeń regularnych (nazywanych przez autora systemu ANTRA wzorcami). Pakiet ten wykorzystuje reguły tłumaczenia wzorcowego, zapisane w formacie XML. Poniżej znajduje się przykład reguły, która odpowiada za tłumaczenie daty, zapisanej w formacie polskim (dd.mm.yyyy) na format angielski (dd/mm/yyyy). Przykład jest zaczerpnięty z pracy [Gintrowicz, 2007]:

```
<pattern active="1">
    <instance>([0-9]{1,2})[\.]([0-9]{1,2})[\.]([0-9]{2,4})</instance>
    <source>([0-9]{1,2})[\.]([0-9]{1,2})[\.]([0-9]{2,4})</source>
    <target>([0-9]{1,2})[\/]([0-9]{1,2})[\/](([0-9]{2,4})</target>
    <format change="1" />
    <orders>
            <order sourceGroup="1" sufix="/" />
            <order sourceGroup="2" sufix="/" />
            <order sourceGroup="3" sufix="" />
    </orders>
</pattern>
```

Wzorzec w węźle *instance* odpowiada za odnalezienie daty w zdaniu wejściowym. Wzorce w węzłach *source* oraz *target* odpowiadają za odnalezienie dat w przykładzie, odpowiednio w zdaniu źródłowym i docelowym. Kiedy daty w zdaniach: wejściowym, źródłowym i docelowym zostaną znalezione, może nastąpić podstawienie. Przy podstawieniu, docelowa data zostanie wygenerowana na zasadach opisanych w węźle *orders*. Przykład tłumaczenia przy użyciu tej reguły:

Zdanie wejściowe (pl): „Konferencja odbędzie się 12.02.2007 roku."

Przykład:
Zdanie źródłowe (pl): „Konferencja odbędzie się 18.07.2002 roku."
Zdanie docelowe (en): „The conference will take place on 18/07/2002."

Zdanie przetłumaczone (en): „The conference will take place on 12/02/2007."

Na pakiet transferu wzorcowego składa się wiele podobnych reguł.

Według badań przeprowadzonych przez autora systemu ANTRA, zastosowanie pakietu transferu wzorcowego w tłumaczeniu przez analogie skutkuje uzyskaniem wysokiej jakości tłumaczeń (w przeprowadzonych testach system ANTRA wykonywał od 72% do 88% poprawnych podstawień na podstawie wzorców). Jednak podstawienia te, choć na ogół poprawne, wykonywane były niezmiernie rzadko - tylko w 0,84% tłumaczonych zdań. Oznacza to, że pakiet ten funkcjonuje poprawnie, jednak nie przyczynia się znacząco do poprawy jakości tłumaczenia przez analogie.

Pakiet transferu słownikowego

Uzupełnieniem pakietu wzorcowego jest pakiet transferu słownikowego. Odpowiada on za transfer tych części zdań, które nie są objęte wzorcami regularnymi. Zalicza się do nich słowa oraz krótkie frazy.

Pakiet opiera swoje działanie na słowniku. Słownik stanowi zbiór par słów lub fraz, z których jedno jest w języku źródłowym, a drugie w docelowym. W założeniu, słownik ma cechować prostota budowy. Przykładowy słownik, wykorzystywany przy tłumaczeniu z języka polskiego na niemiecki, wygląda następująco (przykład zaczerpnięty z pracy [Gintrowicz, 2007]):

słodycze : Süssigkeiten
rowerze : Fahrrad
kwiat : Blume
kwiaty : Blumen
kwiatów : Blumen
piłka : Ball
piłki : Ball
drzwi : Tür
meble : Möbel
szkoła : Schule
nauczyciel : Lehrer
mama : Mutti
matka : Mutter
tata : Vati
ojciec : Pater

Słownik ten składa się wyłącznie z par słów. Należy zwrócić uwagę na następujące cechy charakterystyczne słownika:

- Słownik zawiera osobne wpisy dla różnych słowoform, pochodzących z tego samego leksemu w języku źródłowym (np. *kwiaty : Blumen* oraz *kwiatów : Blumen*).
- Nawet jeśli słowo w języku źródłowym jest wieloznaczne, brane pod uwagę jest tylko jedno jego tłumaczenie – to występujące najczęściej (np. słowo *piłka* zostało przetłumaczone jako *Ball*, a nie jako *kleine Säge* – mała piła).

W słowniku mogą znaleźć się jednak frazy, podobnie jak w poniższym słowniku dla tłumaczenia polsko-angielskiego, w którym obok słów znajduje się jedna fraza (słownik także zaczerpnięty z pracy [Gintrowicz, 2007]):

jeść : eating
słodycze : sweets
jeździć : riding
na rowerze : on the bike

Zostanie teraz przedstawione przykładowe tłumaczenie, wykonane na podstawie powyższego słownika polsko-angielskiego (przykład wykorzystany w [Gintrowicz, 2007]).

Zdanie wejściowe (pl): „Lubię jeść słodycze po lekcjach."

Przykład:
Zdanie źródłowe (pl): „Lubię jeździć na rowerze po lekcjach."
Zdanie docelowe (en): „I like riding on the bike after lessons."

Zdanie przetłumaczone (en): „I like eating sweets after lessons."

Podczas tłumaczenia zostały wykonane następujące kroki:

1. Zostały wykryte różnice pomiędzy zdaniem wejściowym, a źródłowym (zamiast dwóch słów: *jeść* oraz *słodycze*, występują trzy słowa: *jeździć*, *na* oraz *rowerze*).
2. Na podstawie słownika zostały określone następujące odpowiedniości pomiędzy zdaniem źródłowym, a docelowym: *jeździć* – *riding* oraz *na rowerze* – *on the bike*.
3. Na podstawie słownika słowo *jeść* zostało przetłumaczone na *eating*, a słowo słodycze na *sweets*.
4. Przetłumaczone słowa z punktu 3. zostały wstawione do zdania docelowego w miejsce słów *riding on the bike* (przy wykorzystaniu informacji o odpowiedniości między słowami, pozyskanej w punkcie 2.)
5. Zdanie docelowe zostało zwrócone jako wynik tłumaczenia.

Niestety, pakiet transferu słownikowego w systemie ANTRA nigdy nie został dokończony przez autora. Zabrakło przede wszystkim opracowania odpowiednich słowników. Z tego powodu brakuje informacji, w jakim stopniu tak zrealizowany transfer słownikowy w tłumaczeniu przez analogie mógł przyczynić się do poprawy jakości tłumaczenia.

Pakiet oceny jakości tłumaczenia

Zgodnie z ogólną specyfikacją algorytmu tłumaczenia przez analogie, zaprezentowaną w rozdziale 1.1 niniejszej pracy, system ANTRA podaje ocenę jakości wykonywanych przez siebie tłumaczeń. Ocena ta jest liczbą rzeczywistą z przedziału **[0,1]**, najczęściej podawaną w formacie procentowym.

Obliczanie oceny tłumaczenia w systemie ANTRA jest ściśle związane z algorytmem transferu. Zawsze wtedy, kiedy następuje transfer wzorców regularnych lub słów ze zdania wejściowego do docelowego, te same wzorce lub słowa są umieszczane również w zdaniu źródłowym. Na przykład, załóżmy, że system ANTRA wykonuje tłumaczenie przez analogie, przedstawione w przykładzie 1.3.1:

Pamięć tłumaczeń:

pl: „Dzisiaj jest 12.03.2001."

en: „Today is 2001/03/12."

Zdanie wejściowe:

pl: „Dzisiaj jest 4.02.2008."

System przenosi datę 4.02.2008 do zdania docelowego, podając tłumaczenie:

en: „Today is 2008/02/4."

Jednocześnie, przenosi również tę datę do zdania źródłowego z przykładu, produkując zdanie:

pl: „Dzisiaj jest 4.02.2008."

Następnie zdanie to zostaje porównane ze zdaniem źródłowym z przykładu. W tym przypadku zdania te nie różnią się, więc system poda wysoką ocenę tłumaczenia. Ogólnie, jako ocena tłumaczenia podawany jest wynik działania funkcji podobieństwa (opisanej dokładnie w [Gintrowicz, 2007]) na tym ostatnim zdaniu oraz na zdaniu wejściowym.

Autor systemu ANTRA nie podaje, w jakim stopniu tak obliczana ocena tłumaczenia jest skorelowana z oceną tłumaczenia, jaką podałby człowiek. Autor jednak uwagę na fakt, że transfer wzorców i słów ze zdania wejściowego do docelowego różni się od transferu ze zdania wejściowego do źródłowego. W tym pierwszym

musi następować tłumaczenie słów i wzorców, w drugim – nie. Z tego powodu nie sposób ocenić tego, jak dobrze udało się zmodyfikować zdanie docelowe, patrząc tylko na to, jak zostało zmodyfikowane zdanie źródłowe. W praktyce oznacza to, że ocena tłumaczenia w systemie ANTRA może nie być wiarygodna.

4.2.3. Własny wkład w rozwój heTMana

Do zespołu programistycznego, pracującego nad heTManem, dołączyłem w listopadzie 2007 roku. Początkowo zajmowałem się głównie rozwojem interfejsu systemu. Jestem autorem dużej ilości kontrolek i przełączników, które można zobaczyć w obecnej wersji systemu.

Z czasem, zakres moich obowiązków rozszerzył się na dokonywanie zmian w logice systemu (moduł core), opiekę nad modułem WS oraz udział w przygotowywaniu interfejsu do usługi Globalizacji. Wreszcie, po około 4 miesiącach, otrzymałem zadania związane bezpośrednio z lingwistyką komputerową.

Przejąłem moduł splitter. Jestem obecnie odpowiedzialny za utrzymywanie reguł segmentacji (zapisanych w formacie SRX). Problemem jest każda zmiana w regułach segmentacji, gdyż często powoduje ona, że aktualna pamięć tłumaczeń staje się bezużyteczna. Dzieje się tak dlatego, że jeśli na przykład pamięć tłumaczeń została przygotowana w oparciu o reguły często zabraniające podziału zdania, to zawiera przykłady z długimi zdaniami źródłowymi i docelowymi. Zawiera wtedy na przykład zdanie źródłowe:

pl: „Poszedłem na spacer; była piękna pogoda”

Załóżmy wtedy, że zmiany w regułach segmentacji spowodują częstsze dzielenie na zdania (na przykład spowodują dzielenie na zdania po znaku średnika). Niech tekst do przetłumaczenia brzmi:

pl: „Poszedłem na spacer; była brzydka pogoda”.

Zgodnie ze schematem S1, tekst do przetłumaczenia musi podlec segmentacji. W wyniku tej operacji powstaną dwa zdania:

pl: „Poszedłem na spacer;”

pl: „ była brzydka pogoda”.

Zdań tych nie uda się dobrze przetłumaczyć przez analogie, korzystając ze starej pamięci tłumaczeń, zawierającej tylko jedno długie zdanie.

Pracowałem także nad modułem aligner. Moje prace polegały na przejęciu kodów źródłowych od Jarosława Lipskiego oraz opracowaniu nowej funkcji obliczającej prawdopodobieństwo, że dane zdanie docelowe jest tłumaczeniem danego zdania źródłowego. Prawdopodobieństwo to było obliczane w oparciu o długość zdań (jak w algorytmie Gale'a Church'a [Gale, Church, 1991]) oraz o dopasowanie słownikowe poszczególnych słów ze zdania źródłowego do słów ze zdania docelowego.

Jednak moim głównym zadaniem było opracowanie nowego algorytmu tłumaczenia przez analogie. Początkowo miał on powstać jako rozszerzenie modułu ANTRA. Rozszerzenie to miało polegać przede wszystkim na dodaniu funkcjonalności podstawiania słów w przykładzie na podstawie słownika. Oprócz tego, istotne było zwiększenie szybkości działania całego algorytmu tłumaczenia przez analogie.

Z czasem zdecydowano, że wymagania te zostaną zrealizowane w zupełnie nowym algorytmie tłumaczenia przez analogie. Moduł ANTRA został więc zastąpiony modułem o nazwie NeLex.

ROZDZIAŁ 5
NeLex – NOWY SYSTEM TŁUMACZENIA PRZEZ ANALOGIE

5.1. ALGORYTM TŁUMACZENIA PRZEZ ANALOGIE

5.1.1. Specyfikacja wejścia i wyjścia

Nowy algorytm transferu przez analogie wpisuje się w ogólne wytyczne, przedstawione w rozdziale 1.1. Oznacza to, że wejściem do algorytmu są:

- Zdanie wejściowe do przetłumaczenia (nazywane wzorem systemu ANTRA [Gintrowicz, 2007] **srcInstance**)
- Zdanie źródłowe przykładu (nazywane **srcTemplate**)
- Zdanie docelowe przykładu (nazywane **trgTemplate**)

Dodatkowo, ze względu na zależność stosowanych algorytmów od języków (konieczność wykorzystania odpowiednich słowników), konieczne jest podanie języka źródłowego i docelowego przy uruchomieniu systemu.

Wyjściem algorytmu jest zdanie przetłumaczone, nazywane **trgInstance** oraz ocena tłumaczenia, nazywana **score**.

W kolejnych podrozdziałach zostaną omówione etapy pracy nad tłumaczeniem zdania.

5.1.2. Tokenizacja i podział na jednostki leksykalne

Tokenizacja jest to podział zdania na tokeny (pojęcie tokenu zostało zdefiniowane w rozdziale 1.2). Operacje tokenizacji i podziału na jednostki leksykalne są realizowane w krokach 1 – 4 algorytmu. Poniżej został przedstawiony ich ogólny opis słowny.

1. Jeśli **srcInstance** = **srcTemplate**, to zwróć **trgInstance** równe **trgTemplate** oraz **score** równy 100% i zakończ pracę.
2. Podziel **srcTemplate**, **trgTemplate** oraz **srcInstance** na tokeny.
3. Dla każdego ze zdań w punkcie 3. utwórz listę jednostek leksykalnych tego zdania:
    a. Oznacz jednostki nazwane w tym zdaniu.
    b. Dla każdej odnalezionej jednostki nazwanej utwórz osobną jednostkę leksykalną.
    c. Uzupełnij listę jednostek leksykalnych w oparciu o listę tokenów tego zdania.
4. Jeśli listy jednostek leksykalnych zdań **srcInstance** oraz **srcTemplate** są identyczne, zwróć **trgInstance** równy **trgTemplate** oraz **score** równy 99% i zakończ pracę.

Komentarz

Dobrą cechą algorytmu jest to, że nie zajmuje się on transferem zdania, gdy nie ma takiej potrzeby. W przypadku, gdy przykład zostanie dobrany idealnie, tj. **srcInstance** = **srcTemplate**, zostanie natychmiast zwrócony wynik (krok 1). Co więcej, jeśli nawet zdania te nie są równe co do znaku, ale składają się z tych samych jednostek leksykalnych (różnią się jedynie znakami nie wpływającymi na znaczenie zdań, np. spacjami), to system także wykryje taką sytuację i natychmiast zwróci odpowiedni wynik tłumaczenia z oceną 99% (krok 4).

W kroku 3a. wykorzystywany jest mechanizm HNERT, służący do oznaczania jednostek nazwanych, przybliżony w rozdziale 5.2.2.

Krok 3c. polega na utworzeniu jednostek leksykalnych z tokenów, które nie weszły w skład jednostek nazwanych. Operacje wykonywane w tym kroku polegają na przykład na utworzeniu jednostki leksykalnej typu słowo z tokenu będącego ciągiem liter.

5.1.3. Przetwarzanie początkowe

Na przetwarzanie początkowe składają się operacje, które mają wstępnie zmodyfikować **trgTemplate** (który zostanie na końcu zwrócony jako wynik tłumaczenia). Chodzi o takie zmodyfikowanie **trgTemplate**, że:

- **trgTemplate** będzie się zaczynało tą samą sekwencją białych znaków oraz znaczników języka HTML, co **srcInstance**
- **trgTemplate** będzie się kończyło tą samą sekwencją białych znaków oraz znaczników języka HTML, co **srcInstance**

Powyższe operacje wykonywane są w ramach kroków 5 – 7 algorytmu:

5. Przenieś białe znaki
    a. Jeśli **srcInstance** zaczyna się serią białych znaków, a **trgTemplate** nie, dodaj tę serię na początek **trgTemplate**.
    b. Jeśli zarówno **srcInstance**, jak i **trgTemplate** zaczynają się serią białych znaków, zastąp serię z **trgTemplate** serią z **srcInstance**.
    c. Jeśli **trgTemplate** zaczyna się serią białych znaków, a **srcInstance** nie, usuń tę serię z **trgTemplate**
    d. Powtórz kroki a-d dla serii białych znaków na końcach zdań.
6. Przenieś znaczniki HTML
    a. Wykonaj operację analogicznie do kroków 5a-5d
7. Uzupełnij końcową interpunkcję
    a. Jeśli **srcInstance** kończy się znakiem interpunkcyjnym, a **trgTemplate** nie, przenieś znak interpunkcyjny z końca **srcInstance** na koniec **trgTemplate**.

Komentarz

Przetwarzanie początkowe ma jedynie znaczenie kosmetyczne. Staje się jednak istotne w praktycznych zastosowaniach. Na przykład w Globalizacji (rozdział 4.1.5) ważne jest, aby przetłumaczone zdania były otoczone właściwymi białymi znakami oraz znacznikami języka HTML.

5.1.4. Zbudowanie siatki odpowiedniości między zdaniem wejściowym, a źródłowym

Operacja ta jest pierwszym krokiem właściwego transferu przez analogie. Znaczenie siatki odpowiedniości zostało wyjaśnione na przykładach w rozdziale 1.3. Siatka odpowiedniości zawiera informacje o tym, jakie słowa i jednostki nazwane z **srcInstance** będą mogły być podstawione. Konstrukcja siatki odpowiedniości odbywa się w krokach 8 – 11 algorytmu:

8. Dla każdej pary jednostek leksykalnych (**srcTemplateUnit***,* **srcInstanceUnit***)* takiej, że **srcTemplateUnit** należy do **srcTemplate**, a **srcInstanceUnit** do **srcInstance** wyznacz stopień odpowiedniości między nimi (**correspondence**)
    a. Jeśli jednostki są identycznymi znakami interpunkcyjnymi, **correspondence** := 1.0
    b. Jeśli tylko jedna jednostka jest znakiem interpunkcyjnym, **correspondence** := -0.25
    c. Jeśli obie jednostki są identyczne, **correspondence** := 1.0 (identyczne słowa w tym samym języku odpowiadają sobie).
    d. Jeśli obie jednostki leksykalne są jednostkami nazwanymi tego samego typu, **correspondence** := -0.5, jeśli tylko jedna jednostka jest jednostką nazwaną lub typ jednostek nazwanych jest niezgodny: **correspondence** := -0.25
    e. Jeśli przynajmniej jednej jednostki leksykalnej nie ma w słowniku, **correspondence** := -1
    f. Jeśli obie jednostki leksykalne są słowoformami tego samego leksemu **correspondence** := 0.5, jeśli nie – **correspondence** := 0.0
9. Zbuduj reprezentację siatki odpowiedniości w formie tablicy **srcTemplateInstanceArray**
10. Jeśli w jakimś wierszu lub kolumnie znajduje się wartość 0.5 oraz same wartości mniejsze od 0, zmień wartość 0.5 na 1.0
11. Wyzeruj w tej tablicy powtarzające się w wierszach wartości 1.0 oraz -0.5

12. Sprawdź, które znaki przestankowe znajdują się w **srcTemplate**, a brakuje ich w **srcInstance** (będą one przeznaczone do usunięcia)

Komentarz

Podczas tworzenia siatki odpowiedniości obliczane są wartości zmiennej **correspondence**, określającej odpowiedniość między dwoma jednostkami leksykalnymi. Znaczenia wartości zmiennej **correspondence** są objaśnione w poniższej tabeli:

Wartość	**Znaczenie**
-1	Brak wiedzy na temat odpowiedniości
-0.75	Identyczność jednostek nazwanych
-0.5	Odpowiedniość dwóch jednostek nazwanych tego samego typu
-0.25	Brak odpowiedniości między jednostkami leksykalnymi
0	Brak odpowiedniości słownikowej pomiędzy słowami (słowa nie pochodzą z jednego leksemu)
0.5	Odpowiedniość słownikowa pomiędzy słowami (słowa pochodzą z jednego leksemu)
1.0	Odpowiedniość pewna, identyczność

Tabela T7 – wartości zmiennej **correspondence**

Siatka odpowiedniości jest reprezentowana za pomocą tablicy. Poniżej przedstawiona została przykładowa reprezentacja siatki odpowiedniości dla zdań **srcInstance** = *„Dnia 12.02.2009 roku jest brzydka pogoda.”;* **srcTemplate** = *„Dnia 6.03.2009 roku była ładna pogoda.”*

	Dnia	6.03.2009	roku	była	ładna	pogoda	.
Dnia	1.0	-0.25	0.0	0.0	0.0	0.0	-0.25
12.02.2009	-0.25	-0.5	-0.25	-0.25	-0.25	-0.25	-0.25
roku	0.0	-0.25	1.0	0.0	0.0	0.0	-0.25
jest	0.0	-0.25	0.0	0.5	0.0	0.0	-0.25
brzydka	0.0	-0.25	0.0	0.0	0.0	0.0	-0.25
pogoda	0.0	-0.25	0.0	0.0	0.0	1.0	-0.25
.	-0.25	-0.25	-0.25	-0.25	-0.25	-0.25	1.0

Tabela T8 – przykładowa siatka odpowiedniości

Krok 10. ma na celu pozyskanie dodatkowych informacji o odpowiedniości między słowami. Wnioskowanie następuje według następującej zasady: jeśli słowo $\mathbf{s_1}$ odpowiada słownikowo słowu $\mathbf{s_2}$ (oba słowa pochodzą z tego samego leksemu) oraz słowo $\mathbf{s_1}$ nie zostało dopasowane żadnemu innemu słowu ze zdania, z którego

pochodzi $\mathbf{s_2}$, to słowo $\mathbf{s_1}$ jednoznacznie odpowiada słowu $\mathbf{s_2}$. Na przykład dla zdań *„Kiedyś była ładna pogoda”* i *„Kiedyś będzie ładna pogoda”*, siatka odpowiedniości wygląda następująco:

	Kiedyś	będzie	ładna	pogoda
Kiedyś	1.0	0.0	0.0	0.0
była	0.0	0.5	0.0	0.0
ładna	0.0	0.0	1.0	0.0
pogoda	0.0	0.0	0.0	1.0

Tabela T9 – siatka odpowiedniości przed ujednoznacznienie

Jak można zaobserwować, słowa *„była”* i *„będzie”* pochodzą z tego samego leksemu, dlatego ich wartość **correspondence** wynosi 0.5. Jednocześnie słowo *„była”* nie odpowiada żadnemu z pozostałych słów zdania *„Kiedyś będzie ładna pogoda”*, a słowo *„będzie”* nie odpowiada żadnemu z pozostałych słów zdania *„Kiedyś była ładna pogoda”.* W związku z tym, wartość **correspondence** dla słów *„była”* i *„będzie”* przyjmie w tym przypadku wartość 1.0.

Krok 11. ma na celu uzyskanie jednoznaczności odpowiedniości. Gdy na przykład w wierszu znajdą się dwie wartości -0.5, oznacza to, że jedna jednostka nazwana z **srcInstance** odpowiada dwóm jednostkom nazwanym z **srcTemplate**. Po wykonaniu kroku 11., druga wartość -0.5 zostanie wyzerowana. Tym samym, jednostka nazwana z **srcInstance** będzie odpowiadała tylko pierwszej jednostce nazwanej z **srcTemplate** (jest to ujednoznacznienie polegające na arbitralnym wyborze pierwszej odpowiedniości).

5.1.5. Znalezienie słów i jednostek nazwanych do przeniesienia

Kolejne kroki algorytmu mają na celu znalezienie tych słów i jednostek nazwanych, które zostaną przeniesione z **srcInstance** do **trgTemplate**. W tej fazie algorytmu równolegle rozpoczyna się również zbieranie danych, koniecznych do obliczenia oceny tłumaczenia. W tym celu definiowana jest zmienna **discrepancies**, która będzie przechowywała wartość liczbową, odpowiadającą ilości rozbieżności pomiędzy **srcInstance**, a **srcTemplate**. Wszystko to jest wykonywane w krokach 13 – 16 algorytmu.

13. **discrepancies** := 0
14. Znajdź puste wiersze i kolumny tablicy **srcTemplateInstanceArray** (tj. takie, w których są tylko wartości 0, -0.25 lub -1). Podczas przeszukiwania:
    a. Dla każdej znalezionej w tablicy wartości **correspondence** = 0.5: **discrepancies** := **discrepancies** + 0.5
    b. Dla każdej znalezionej w tablicy wartości **correspondence** = -0.5: **discrepancies** := **discrepancies** + 0.5
    c. Dla każdej jednostki nazwanej, która występuje w **srcTemplate**, ale nie w **srcInstance**:
    discrepancies := **discrepancies** + 1.5
    d. Dla każdej jednostki nazwanej, która występuje w **srcInstance**, ale nie w **srcTemplate**:
    discrepancies := **discrepancies** + 1.5
    e. Zwiększ **discrepancies** o 0.25 na zasadzie analogicznej do punktów c. i d., dla każdego niedopasowanego znaku interpunkcyjnego.
    f. Zwiększ **discrepancies** o 1.0 na zasadzie analogicznej do punktów c. i d., dla każdego niedopasowanego słowa.
15. Utwórz listę par słów do podmiany
    a. Weź wszystkie pary **(m,n)**, takie że **m** jest numerem pustego wiersza w **srcTemplateInstanceArray**, a **n** – numerem pustej kolumny w tej tablicy
    b. Utwórz **substArray** jako pustą listę par liczb.
    c. Dla każdej pary **(m,n)** z punktu a.:
        i. Jeśli **srcTemplateInstance[m,n]** = 0 oraz **substArray** nie zawiera pary o poprzedniku równym **m**, ani o następniku równym **n**, dodaj parę **(m,n)** do **substArray** oraz **srcTemplateInstance[m,n]** := 1.0
        ii. Jeśli **srcTemplateInstance[m,n]** = -0.5 oraz **substArray** nie zawiera pary o poprzedniku równym **m**, ani o następniku równym **n**, dodaj parę **(m,n)** do **substArray** oraz **srcTemplateInstance[m,n]** := 1.0
16. Uwzględnij inwersje (zamiany kolejności słów) w **srcTemplateInstance**
    a. Policz ilość inwersji **inversions** w **srcTemplateInstance**
        i. **inversions** := 0

ii. Dla każdego wiersza o numerze **w** znajdź pozycję pierwszej wartości 1.0 i oznacz tę pozycję jako $\mathbf{p_w}$

iii. Dla każdej pary wierszy $\mathbf{w_1}$, $\mathbf{w_2}$, takiej że $\mathbf{w_1} < \mathbf{w_2}$ oraz $\mathbf{p_1} > \mathbf{p_2}$: **inversions** := **inversions** + 1.0

b. **discrepancies** := **discrepancies** + (**inversions** / 2)

Komentarz

Pusty wiersz w tablicy **srcTemplateInstance** wskazuje na jednostkę leksykalną, która znalazła się w **srcInstance**, ale nie została dopasowana do żadnej jednostki leksykalnej z **srcTemplate**. Pusta kolumna natomiast wskazuje na jednostkę leksykalną, która znalazła się w **srcTemplate**, ale nie została dopasowana do żadnej jednostki leksykalnej z **srcInstance**. Przecięcie pustego wiersza i pustej kolumny wskazuje na parę słów lub jednostek nazwanych, które nie zostały dopasowane do żadnych słów lub jednostek nazwanych z przeciwnego zdania. Jeśli wartość w tablicy w miejscu takiego przecięcia jest równa -0.5, mamy do czynienia z dwoma jednostkami nazwanymi, jeśli natomiast wartość w tym miejscu wynosi 0.0 – są to dwa słowa, o których wiemy, że na pewno nie pochodzą z tego samego leksemu. To właśnie takie pary jednostek leksykalnych staną się miejscami podmiany. Mechanizm tej podmiany zostanie wyjaśniony w rozdziale 5.1.7.

Co istotne, dopasowanie jednostek leksykalnych musi być jednoznaczne, aby było wiadomo co należy podstawiać. Aby uzyskać takie dopasowanie, wykonuje się operację ujednoznacznienia. Bierze ona pod uwagę pierwsze możliwe dopasowanie. Na przykład, jeśli w **srcTemplateInstance** puste są wiersze 3 i 5 oraz kolumny 4 i 6, do **substArray** zostaną dodane pary (3,4) oraz (5,6) ale już nie (3,6) i (5,4).

5.1.6. Zbudowanie siatki odpowiedniości pomiędzy zdaniem źródłowym, a docelowym

Kolejną fazą algorytmu transferu jest zbudowanie tablicy **srcTrgArray**, reprezentującej siatkę odpowiedniości pomiędzy **srcTemplate** i **trgTemplate**. Tablica ta będzie konieczna podczas przenoszenia jednostek leksykalnych z **srcInstance** do **trgTemplate**.

Konstrukcja tablicy **srcTrgArray** jest bardzo podobna do konstrukcji tablicy **srcTemplateInstanceArray**. Tworzenie tablicy **srcTrgArray** następuje w krokach 17 – 21 algorytmu.

17. Dla każdej pary jednostek leksykalnych (**srcTemplateUnit***,* **trgTemplateUnit***)* takiej, że **srcTemplateUnit** należy do **srcTemplate**, a **trgTemplateUnit** do **trgTemplate** wyznacz stopień odpowiedniości (**correspondence**)
    a. Jeśli jednostki są identycznymi znakami interpunkcyjnymi, **correspondence** := 1.0
    b. Jeśli tylko jedna jednostka jest znakiem interpunkcyjnym, **correspondence** := -0.25
    c. Jeśli obie jednostki są jednostkami nazwanymi tego samego typu, **correspondence** := -0.5, jeśli tylko jedna jednostka jest jednostką nazwaną lub typ jednostek nazwanych jest niezgodny: **correspondence** := -0.25
    d. Jeśli jednostki docelowej nie ma w słowniku, **correspondence** := -1
    e. W pozostałych przypadkach przyjmij, że obie jednostki leksykalne są słowami i oblicz stopień odpowiedniości pomiędzy słowami w dwóch językach:
        i. Wyznacz listę leksemów w języku docelowym, na które może zostać przetłumaczone słowo źródłowe.
        ii. Wyznacz listę słowoform w języku docelowym, odpowiadających leksemom z poprzedniej listy.
        iii. Jeśli lista ta jest pusta (co następuje wtedy, gdy słowa w języku źródłowym nie ma w słowniku), **correspondence** := -1
        iv. W przeciwnym przypadku, jeśli na tej liście znajduje się słowo docelowe, **correspondence** := 0.5, jeśli nie: **correspondence** := 0.0
18. Zbuduj reprezentację siatki odpowiedniości w formie tablicy **srcTrgArray**
19. Jeśli w jakimś wierszu lub kolumnie znajduje się wartość 0.5 oraz same wartości mniejsze od 0, zmień wartość 0.5 na 1.0
20. Wyzeruj w tej tablicy powtarzające się w wierszach wartości -0.5
21. Uwzględnij identyczność jednostek nazwanych
    a. Wyzeruj w tej tablicy powtarzające się w wierszach wartości -0.75

b. Jeśli w jakimś wierszu znajduje się wartość -0.75, zamień ją na -0.5, a pozostałe wartości -0.5 w tym wierszu zamień na 0.0

Komentarz

Najistotniejszą różnicą, w stosunku do budowania tablicy **srcTemplateInstanceArray** jest sposób obliczania **correspondence** dla dwóch słów. Wykorzystuje on informacje słownikowe o:

- przynależności danych słów do leksemów
- możliwości tłumaczenia danego leksemu w języku źródłowym na inny dany leksem w języku docelowym

Słownik jest dokładniej opisany w rozdziale 5.2.1.

Poza tym, istotny jest krok 21., który polega na uwzględnieniu identyczności jednostek nazwanych. Znaczenie działań w kroku 21. ilustruje następujący przykład:

srcTemplate (pl): „Wakacje trwały od 26.06.2003 do 1.09.2003."
trgTemplate (en):"Holiday ended on 1.09.2003 and began on 26.06.2003"

srcInstance (pl):"Wakacje trwały od 29.06.2005 do 2.09.2005"

Gdyby algorytm nie miał wiedzy na temat identyczności jednostek nazwanych, przyporządkowałby datę 26.06.2003 z **srcTemplate** do 1.09.2003 z **trgTemplate**, a 1.09.2003 z **srcTemplate** do 26.06.2003 z **trgTemplate**. Jak wynika z przytoczonych w rozdziale 5.1.7 kroków algorytmu tłumaczącego, skutkowałoby to wyprodukowaniem błędnego tłumaczenia zdania:

trgInstance (en):"Holiday ended on 09/02/2005 and began on 06/29/2005"

Jednak algorytm posiada wiedzę na temat identyczności dat 26.06.2003 oraz 1.09.2003 w **srcTemplate** i **trgTemplate**, dzięki czemu potrafi zbudować poprawną siatkę odpowiedniości.

5.1.7. Zmodyfikowanie zdania docelowego celem wyprodukowania tłumaczenia

Kolejne kroki algorytmu (22 – 25) są jego sednem. Podczas nich dochodzi do faktycznego transferu i wyprodukowania wyniku tłumaczenia. Oprócz tego, dokonywane są obliczenia konieczne do podania końcowej oceny tłumaczenia.

22. Dla każdej pary **(i,j)** z **substArray**
    a. Pobierz jednostkę leksykalną **srcInstanceUnit** z pozycji **i** z **srcInstance**.
    b. Jeśli w **j**-tej kolumnie **srcTrgArray** jest wartość 1.0 lub -0.5, pobierz **k** – numer wiersza, w którym ta wartość się znajduje
    c. Jeśli odszukana wartość wynosi 1.0, to przetłumacz słowo **srcInstanceUnit** na język docelowy:
        i. Znajdź leksem tego słowa w języku źródłowym
        ii. Znajdź leksem tego słowa w języku docelowym (korzystając ze słownika)
        iii. Jako tłumaczenie przyjmij pierwszą słowoformę z leksemu w języku docelowym
    d. W pozostałych przypadkach, jeśli odszukana wartość wynosi -0.5, to przetłumacz jednostkę nazwaną **srcInstanceUnit** na język docelowy (zgodnie ze specyficznymi dla konkretnych typów jednostek nazwanych regułami tłumaczenia).
    e. Wstaw przetłumaczone **srcInstanceUnit** na miejsce **k** w **trgTemplate**
23. Policz ilość podstawień słownikowych i umieść ją w zmiennej **dictSubst**
24. Policz ilość podstawień jednostek nazwanych i umieść ją w zmiennej **neSubst**
25. **discrepancies** := **discrepancies** - (**dictSubst** * 1.5) - (**neSubst***0.4)

Komentarz

Podczas przenoszenia jednostek leksykalnych z **srcInstance** do **trgTemplate** zostały wykorzystane obie przygotowane wcześniej siatki odpowiedniości.

Każde udane przeniesienie powoduje zmniejszenie zmiennej **discrepancies**, która określa rozbieżności między **srcInstance**, a **srcTemplate**. Przy każdym

udanym podstawieniu słownikowym zmienna **discrepancies** jest zmniejszana o wartość 1.5. Dzieje się tak dlatego, że podczas określania odpowiedniości pomiędzy **srcInstance**, a **srcTemplate**, brak odpowiedniości pary słów spowodował zwiększenie zmiennej **discrepancies** o 2.0 (po 1.0 za nie odnalezienie każdego ze słów w przeciwnym zdaniu). Zmniejszenie zmiennej o 1.5 za każde udane podstawienie powoduje częściowe wyrównanie poprzednio naliczonej wartości rozbieżności.

W przypadku podstawień jednostek nazwanych dokonywany jest podobny zabieg. Jeśli wcześniej została znaleziona para odpowiadających sobie (a nie identycznych) jednostek nazwanych w **srcInstance** i **srcTemplate**, zmienna **discrepancies** została zwiększona o 0.5 (z powodu doliczania do **discrepancies** wartości 0.5 za każdą wartość -0.5 w tablicy **srcTemplateInstanceArray** – krok 14b. algorytmu). Po udanym podstawieniu jednostki nazwanej, poprzednia zmiana wartości zmiennej **discrepancies** została niemal całkowicie zniwelowana (zmienna **discrepancies** została zmniejszona o 0.4 po wcześniejszym zwiększeniu o 0.5). Oznacza to, że podstawienia jednostek nazwanych bardzo nieznacznie zmniejszają ocenę tłumaczenia zdania. Na przykład, przy tłumaczeniu zdania: *„Dzisiaj jest 1.02.2001”* na język angielski, z wykorzystaniem przykładu:

pl: „Dzisiaj jest 23.05.2004”
en: „Today is 23.05.2004”

Podawane jest tłumaczenie: *„Today is 2001-02-01”* z oceną 96.67 %.

5.1.8. Obliczenie oceny tłumaczenia i zakończenie działania algorytmu

Algorytm kończy obliczenie oceny tłumaczenia i zwrócenie wyniku:

26. Oblicz **avgLength** – średnią ilość jednostek leksykalnych w zdaniach **srcInstance** i **srcTemplate**.
27. **score** := 1 - (**discrepancies** / **avgLength**)
28. **trgInstance** := **trgTemplate** (**trgTemplate** zostało wcześniej zmodyfikowane)
29. Zwróć **trgInstance** i **score**.

5.2. STRUKTURA SYSTEMU I MODUŁY POMOCNICZE

System NeLex ma budowę modułową. Każdy z modułów wchodzących w skład systemu posiada określoną odpowiedzialność. Poniżej znajduje się wykaz najważniejszych modułów systemu.

Nazwa modułu	**Odpowiedzialność**
NeLexTranslator	Wykonywanie tłumaczenia i obliczanie jego oceny (główny moduł systemu, korzystający z pozostałych)
SentenceProcessor	Tokenizacja zdań i przetwarzanie początkowe
AlignmentArrayBuilder	Tworzenie siatek powiązań pomiędzy zdaniami i dostarczanie danych do obliczenia oceny tłumaczenia
WordProcessor	Zarządzanie słownikami oraz wyszukiwanie słów w słownikach
HNERT	Oznaczanie jednostek nazwanych w zdaniach

Tabela T10 – odpowiedzialności modułów systemu NeLex

Działanie modułów NeLexTranslator, SentenceProcessor oraz AlignmentArrayBuilder zostało opisane dokładnie w paragrafie 5.1. W paragrafie 5.2 zostanie przybliżone działanie modułów WordProcessor oraz HNERT.

5.2.1. Moduł WordProcessor

Jak już zostało wspomniane, moduł ten zajmuje się wykonywaniem operacji przy użyciu słowników. Dla określonego kierunku tłumaczenia (np. z języka polskiego na angielski) moduł ładuje trzy słowniki:

- Słownik słowoform źródłowych
- Słownik słowoform docelowych
- Słownik tłumaczeń leksemów

Słownik słowoform źródłowych stanowi lista par **(w,i)**, gdzie **w** jest słowoformą w języku źródłowym, a **i** – identyfikatorem leksemu, z którego pochodzi słowoforma **w**. Poniżej znajduje się przykładowy słownik słowoform źródłowych dla języka polskiego:

dom	5
domu	5
piłka	13
piłce	13
piłek	13
zamek	2
zamku	2
zamka	2
zamek	3

Słownik słowoform źródłowych

W powyższym przykładzie, słowoforma *„zamek”* może pochodzić z dwóch różnych leksemów. Jednym jest zamek, oznaczający zamek w drzwiach, drugim natomiast zamek – budowla.

Słownik słowoform docelowych jest strukturą analogiczną do słownika słowoform źródłowych – jedyną różnicą jest język słowoform. Oto przykład takiego słownika, dla języka angielskiego:

house	1
houses	1
castle	2
lock	4
lock	6
locking	6
ball	7

Słownik słowoform docelowych

Na powyższym przykładzie widać, że słowoforma lock może pochodzić z dwóch leksemów. Może bowiem oznaczać rzeczownik lock, czyli zamek u drzwi lub czasownik to lock, czyli zamykać.

Ostatnim ze słowników jest słownik tłumaczeń leksemów. Składa się on z par **(s,t)**, takich że **s** jest identyfikatorem leksemu w języku źródłowym (nazywanego krócej leksemem źródłowym), a **t** jest identyfikatorem leksemu w języku docelowym (nazywanego leksemem docelowym). Fakt, że w słowniku znajduje się para

(s_0,t_0) jest równoznaczny z tym, że słowoformy w języku źródłowym, należące do leksemu źródłowego o identyfikatorze **s_0** mogą być tłumaczone na słowoformy w języku docelowym, należące do leksemu docelowego o identyfikatorze **t_0**. Poniżej znajduje się słownik tłumaczeń leksemów dla wcześniej przytoczonych słowników słowoform źródłowych i docelowych:

5	1
13	7
2	4

Słownik tłumaczeń leksemów

Na podstawie powyższych słowników, moduł WordProcessor może wykonywać następujące operacje:

- Sprawdzenie, czy dane dwie słowoformy w języku źródłowym mogą pochodzić z tego samego leksemu
- Sprawdzenie, czy dana słowoforma w języku źródłowym może odpowiadać danej słowoformie w języku docelowym
- Podanie tłumaczenia słowoformy w języku źródłowym na język docelowy

Operacja sprawdzenia, czy dane dwie słowoformy w języku źródłowym mogą pochodzić z tego samego leksemu, jest wykonywana w sposób następujący:
Dane są słowoformy **w_1** i **w_2** w języku źródłowym. Wykonywane są kroki:

1. Zgromadź listę **L_1** wszystkich takich liczb **i_1**, że **(w_1, i_1)** należy do słownika słowoform źródłowych.
2. Zgromadź listę **L_2** wszystkich takich liczb **i_2**, że **(w_2, i_2)** należy do słownika słowoform źródłowych.
3. Jeśli listy **L_1** i **L_2** mają przynajmniej jeden element wspólny, zwróć **true**, w przeciwnym wypadku zwróć **false**.

Operacja sprawdzenia, czy dana słowoforma w języku źródłowym może odpowiadać danej słowoformie w języku docelowym jest wykonywana w sposób następujący:
Dane są: słowoforma **w_s** w języku źródłowym oraz słowoforma **w_t** w języku docelowym. Wykonywane są kroki:

1. Zgromadź listę **L_s** wszystkich takich liczb **i_s**, że **(w_s, i_s)** należy do słownika słowoform źródłowych.

2. Zgromadź listę $\mathbf{L_t}$ wszystkich takich liczb $\mathbf{i_t}$, że $\mathbf{(i_s, i_t)}$ należy do słownika tłumaczeń leksemów, dla pewnego $\mathbf{i_s}$, należącego do $\mathbf{L_s}$.
3. Zgromadź listę $\mathbf{L_{wt}}$ wszystkich takich słowoform **w** w języku docelowym, że $\mathbf{(w, i_t)}$ należy do słownika słowoform docelowych dla jakiegoś $\mathbf{i_t}$, należącego do $\mathbf{L_t}$.
4. Jeśli $\mathbf{w_t}$ należy do $\mathbf{L_{wt}}$, zwróć **true**, w przeciwnym wypadku zwróć **false**.

Operacja przetłumaczenia słowoformy w języku źródłowym na język docelowy wykonywana jest następująco:

Dana jest słowoforma $\mathbf{w_s}$ w języku źródłowym. Wykonywane są kroki:

1. Znajdź pierwszą liczbę $\mathbf{i_s}$ taką, że $\mathbf{(w_s, i_s)}$ należy do słownika słowoform źródłowych.
2. Znajdź pierwszą liczbę $\mathbf{i_t}$ taką, że $\mathbf{(i_s, i_t)}$ należy do słownika tłumaczeń leksemów.
3. Zwróć pierwszą słowoformę $\mathbf{w_t}$ taką, że $\mathbf{(w_t, i_t)}$ należy do słownika słowoform docelowych.

W związku z tym, podczas tłumaczenia słowa z języka źródłowego na docelowy, brane jest pod uwagę tylko pierwsze napotkane tłumaczenie.

5.2.2. Moduł HNERT

Moduł HNERT służy do oznaczania jednostek nazwanych w zdaniach: wejściowym, źródłowym i docelowym. Dodatkowo moduł wyznacza, jak jednostka nazwana powinna być przetłumaczona.

Moduł HNERT opiera swoje działanie o reguły oznaczania i tłumaczenia jednostek nazwanych, zapisane przy użyciu formalizmu NERT (opisanego w [NERT, 2008]). Jednak ze względu na to, że tłumaczenie przez analogie nie przeprowadza głębokiej analizy tekstu, w regułach NERT dla systemu NeLex można obecnie stosować jedynie dopasowania oparte na wzorcach regularnych. Poniżej zamieszczono specyfikację podzbioru formalizmu NERT, używanego przez moduł HNERT (na podstawie formalizmu opisanego w [NERT, 2008]:

```
::nert_file::
        (definition)*                                   # zbiór definicji
        (rule)+                                         # niepusty zbiór reguł

::definition::
        Name =  pattern                                 # definicja makra

::rule::
        Match: pattern                                  # wzorzec dopasowania
        Action: replace (newText [[[;Num ]; Num ])      # akcja tłumaczenia
        Class: Text                                     # typ jednostki nazwanej
::pattern::
        <regExp>(<regExp>)*             # wzorzec dopasowania jest ciągiem wyrażeń regularnych,
                                                        ujętych w znaki < i >.
::newText::
        expression( expression)*

::expression::
        Text |                                          # nowy tekst w tłumaczeniu
        derived                                         # tekst pobrany ze zdania źródłowego

::derived::
        \Num            # wstaw Num-ty element pobrany ze zdania źródłowego
```

Poniżej zostanie przedstawiony przykład reguły oznaczania jednostek nazwanych, zapisanej w formalizmie NERT, przeznaczonej do wykorzystania przez system NeLex:

```
Match: < \w+> < Sp\. z o\.o\.>
Action: replace(\1 Ltd.)
Class: company
```

Reguła składa się z trzech członów: **Match**, **Action** oraz **Class**. Człon **Match** zawiera wzorzec regularny, zapisany w języku NERT, przeznaczony do odnalezienia jednostki nazwanej w zdaniu. W powyższym przypadku, wzorzec regularny z **Match** zostanie przekształcony na język wyrażeń regularny w następujący sposób:

```
(\w+) (Sp\. z o\.o\.)
```

W członie **Action** zakodowana jest reguła tłumaczenia jednostki nazwanej. W powyższym przypadku, reguła pozostawia słowo rozpoznane przez pierwszą grupę, czyli (\w+), bez zmian i zamienia końcówkę *„Sp. z o.o.”* na *„Ltd.”*.

Człon **Class** pozwala na określenie typu jednostki nazwanej. Informacja o typie jednostki nazwanej jest wykorzystywana w krokach 8d. oraz 17c. algorytmu

transferu w systemie NeLex. Bez tej informacji, system mógłby wykonywać błędne podstawienia, na przykład wstawiać datę w miejsce, gdzie powinna być nazwa firmy.

Podsumowując, powyższa reguła rozpoznaje dowolne słowo, po którym następuje napis „Sp. z o.o.” jako bazową nazwę firmy. Tłumaczenie pełnej nazwy firmy polega na pozostawieniu bazowej nazwy firmy bez zmian oraz na zamianie końcówki „Sp. z o.o.” na „Ltd.”. Jest to reguła tłumaczenia nazw firm z języka polskiego na język angielski.

Rozważmy jeszcze jeden przykład reguły rozpoznawania jednostek nazwanych:

Match: <Dz\.\s*U.>\s*<[Nn][Rr]>\s+{NUM},\s*<poz\.>\s*{NUM}
Action: replace(Journal of Laws No. \3, item \5)
Class: JOL

Jest to reguła, potrafiąca odnaleźć w tekście napisy podobne do:
Dz. U. Nr 45, poz. 5

Powyższy przykładowy napis zostanie w myśl tej reguły przetłumaczony na:
Journal of Laws No. 45, item 5

Jest to zatem reguła tłumaczenia odnośników do Dziennika Ustaw z języka polskiego na język angielski.

ROZDZIAŁ 6
AUTORSKIE ULEPSZENIA TŁUMACZENIA TEKSTÓW PRAWNICZYCH

6.1. MOTYWACJA SKUPIENIA SIĘ NA TEKSTACH PRAWNICZYCH

Podczas opracowywania systemu tłumaczenia przez analogie, przedstawionego w niniejszej pracy, szczególny nacisk został położony na jego przydatność do tłumaczenia tekstów prawniczych. Choć system NeLex może z powodzeniem być wykorzystywany do tłumaczenia dowolnych tekstów, najlepsze wyniki uzyskuje dla tekstów prawniczych.

Takie wyspecjalizowanie – wytworzenie systemu tłumaczącego, dedykowanego do tłumaczenia tylko wąskiej klasy tekstów – jest podejściem, które ma szansę stać się przyszłością tłumaczenia automatycznego. Założenie, że tekst pochodzi z określonej dziedziny, w znaczącym stopniu pomaga podczas tłumaczenia.

Założenie to może pomagać w ujednoznacznieniu słów w tekście. Na przykład, słowo *artykuł* może w języku polskim oznaczać m.in.:

- Artykuł w gazecie (dziennikarstwo; *„Przeczytałem ostatnio ciekawy artykuł o pracy za granicą”*)
- Przedmiot (handel; *„Artykuły gospodarstwa domowego”*)
- Część ustawy (prawo; *„Artykuł 3b niniejszej ustawy”*)

Jeśli słowo *artykuł* wystąpi w tekście do tłumaczenia, przy czym wiadomo, że tekst ten jest tekstem prawniczym, z dużym prawdopodobieństwem można przyjąć, że słowo *artykuł* oznacza w tym kontekście część ustawy.

Poza tym, skupienie się na wąskiej klasie tekstów pozwala na opracowanie lepszej jakości słowników, wykorzystywanych podczas tłumaczenia. Wiedząc, że tłumaczone będą tylko teksty prawnicze, można opracować specjalny słownik, zawierający większość słów i sformułowań prawniczych. Jest to możliwe z dwóch zasadniczych powodów. Pierwszym jest fakt, że język prawniczy zawiera znacznie mniej słów i sformułowań niż język, którym posługujemy się na co dzień. Jeśli można opra-

cować wysokiej jakości słownik ogólnego użytku, to tym bardziej możliwe jest opracowanie dobrego słownika, obejmującego węższy zakres słownictwa. Drugim powodem jest lepsze sformalizowanie języka prawniczego. Większość słów i sformułowań prawniczych nie jest wieloznaczna, gdyż teksty prawnicze nie mogą być wieloznaczne.

Wymienione cechy języków specjalistycznych, a w szczególności języka prawniczego, sprawiają, że mogą one z dużym powodzeniem być tłumaczone automatycznie. Stąd, w niniejszej pracy, skupiam się na tłumaczeniu przez analogie tekstów prawniczych.

6.2. WYKORZYSTANIE *NE*

W przypadku systemu NeLex, przyjęcie założenia, że tłumaczone będą teksty prawnicze, pozwoliło na wykonanie następującego usprawnienia: opracowania reguł oznaczania jednostek nazwanych, charakterystycznych dla tekstów prawniczych. Reguły te są wykorzystywane przez moduł HNERT systemu NeLex (opisany w paragrafie 5.2.2).

Zbiór reguł został opracowany w ramach niniejszej pracy magisterskiej. Składa się na niego w sumie 176 reguł, po 22 dla następujących kierunków tłumaczenia:

- Język polski na język angielski
- Język angielski na język polski
- Język polski na język niemiecki
- Język niemiecki na język polski
- Język polski na język rosyjski
- Język rosyjski na język polski
- Język polski na język hiszpański
- Język hiszpański na język polski

W poniższej tabeli zaprezentowano typy jednostek nazwanych, rozpoznawanych przez zbiór reguł.

Lp.	Typ jednostki nazwanej	Opis typu	Przykład (w języku polskim)
1.	JOLFull	Odwołanie do Dziennika Ustaw z konkretnego dnia	Dz. U. z 12.09.2001 Nr 125, poz. 4
2.	JOLYearOnly	Odwołanie do Dziennika Ustaw z konkretnego roku	Dz. U. z 2001r. Nr 125, poz. 4
3.	JOLWithLaterChan-ges	Odwołanie do Dziennika Ustaw „z późniejszymi zmianami”	Dz. U. Nr 125, poz. 4 z późn. zm.
4.	JOL	Odwołanie do Dziennika Ustaw	Dz. U. Nr 125, poz. 4
5.	JOL2	Alternatywna postać odwołania do Dziennika Ustaw	Dz. U. L 125 z 12.09.2001 str. 4
6.	DateNoItem	Odwołanie do części ustawy z konkretnej daty	z 12.09.2001 Nr 125, poz. 4
7.	YearNoItem	Odwołanie do części ustawy z konkretnego roku	z 2001r. Nr 125, poz. 4
8.	NoItem	Odwołanie do części ustawy	Nr 125, poz. 4
9.	Date	Data	12.04.2001
10.	CompanyLtd	Nazwa firmy, będącej spółką z ograniczoną odpowiedzialnością	ACME Sp. z o.o.
11.	CompanyPlc	Nazwa firmy, będącej spółką akcyjną	ACME S.A.
12.	Email	Adres e-mail	imie-nazwisko@example.com
13.	Paragraph	Odwołanie do	§34 ust. 43 pkt g)

		paragrafu	
14.	ParagraphPoint	Odwołanie do ustępu paragrafu	ust. 43 pkt g)
15.	ParagraphSubPoint	Odwołanie do punktu paragrafu	pkt g)
16.	ArtMultipleSec	Odwołanie do dwóch ustępów artykułu	art. 12b ust. 4 i 5
17.	ArtSec	Odwołanie do ustępu artykułu	art. 12b ust. 4
18.	Art	Odwołanie do artykułu	art. 12b
19.	SecPoint	Odwołanie do ustępu i punktu	ust. 4 pkt. 2
20.	Sec	Odwołanie do ustępu	ust. 4
21.	HtmlEntity	Encja języka HTML	&#39
22.	MathNumber	Liczba rzeczywista	121,87

Tabela T11 – typy rozpoznawanych jednostek nazwanych

ROZDZIAŁ 7
PREZENTACJA I OMÓWIENIE WYNIKÓW IMPLEMENTACJI

7.1. TESTY JAKOŚCI TŁUMACZENIA

7.1.1. Przygotowanie testów

Testy jakości tłumaczenia są najistotniejszymi spośród testów systemu NeLex. To one wykazują, w jakim stopniu system potrafi poprawić jakość tłumaczenia.

Aby obiektywnie ocenić system tłumaczący, najlepiej przetłumaczyć przy jego pomocy duży zbiór tekstów. Taki zbiór nazywa się **korpusem** tekstów. W testach systemu NeLex wykorzystano korpus JRC-Acquis (opisany na [JRC]). Jest to korpus tekstów prawniczych Unii Europejskiej, gromadzonych od 1958 roku. Jest dostępny w 22 językach. Teksty są podzielone na zdania oraz dostępne są dopasowania dowolnych dwóch wersji językowych. Dzięki temu możemy otrzymać **korpusy dwujęzyczne**, tj. zbiory par zdań w dwóch różnych językach takich, że drugie zdanie jest tłumaczeniem pierwszego. Liczba par zdań w takich korpusach wynosi około 1200000.

System NeLex pełni wyłącznie rolę modułu tłumaczenia przez analogie (schemat S1). Wejściem do systemu jest zdanie do przetłumaczenia oraz odnaleziony przykład. Wobec tego, aby system NeLex mógł przetłumaczyć dokument, konieczne jest wykorzystanie zewnętrznego modułu, który potrafi odnaleźć odpowiedni przykład w pamięci tłumaczeń. Modułem tym jest system o nazwie MAT ([MAT]).

Przeprowadzenie testu tłumaczenia w konkretnym kierunku składa się z następujących kroków:

1. Wczytanie do pamięci tłumaczeń korpusu dwujęzycznego JRC w odpowiednich językach

2. Losowy wybór ok. 7000-11000 par zdań z powyższego korpusu i umieszczenie ich w zbiorze **T**
3. Dla każdej pary zdań **p**, należącej do **T**:
    a. Przetłumaczenie pierwszego zdania pary **p** z wykorzystaniem pamięci tłumaczeń z usuniętą parą **p** (bez usuwania pary **p** tłumaczenie przez analogie byłoby trywialne).
    b. Porównanie tłumaczenia z drugim zdaniem pary **p** i podanie oceny tłumaczenia (liczby rzeczywistej z zakresu [0,1], zewnętrznej miary jakości tłumaczenia)
4. Obliczenie średniej wszystkich ocen (parametr **avg**)
5. Obliczenie średniej ocen większych od 0 (parametr **nonZeroAvg**)

Krok 3b polega na porównaniu zdania przetłumaczonego automatycznie ze zdaniem przetłumaczonym przez człowieka. Na tej podstawie można dokonać oceny automatycznego tłumaczenia zdania. Czynność ta jest wykonywana automatycznie przy pomocy metryki METEOR (opisanej na [METEOR]).

Podawane są dwa parametry oceny systemu tłumaczącego przez analogie. Parametr **avg** jest średnią wszystkich ocen tłumaczenia. Daje on ogólne pojęcie o jakości tłumaczenia, wykonanego przez system. Średnią tą zaniżają jednak zerowe oceny zdań, dla których tłumaczenie przez analogie nie powiodło się, z powodu braku odpowiedniego przykładu w pamięci tłumaczeń. Wobec tego, bardziej istotny jest drugi parametr – **nonZeroAvg**, czyli średnia ocen tłumaczenia większych od 0. Daje on wyobrażenie o tym, jak wysoka była jakość tłumaczenia zdań, dla których został znaleziony odpowiedni przykład w pamięci tłumaczeń.

Dla każdego kierunku tłumaczenia przeprowadzane są testy typu *baseline*. Najpierw wykonywane jest tłumaczenie zdań ze zbioru **T** przy użyciu najprostszego możliwego modułu tłumaczenia przez analogie, nazywanego właśnie *baseline*, lub punktem odniesienia. Rolę takiego modułu pełni system, który na wejściu otrzymuje przykład oraz zdanie wejściowe i natychmiast zwraca zdanie docelowe (bez żadnych modyfikacji). Następnie, na tym samym zbiorze **T** wykonywane jest tłumaczenie systemem NeLex.

7.1.2. Tłumaczenie polsko-angielskie

Wyniki dla tłumaczenia z języka polskiego na język angielski:

Liczba zdań do przetłumaczenia (moc zbioru **T**) : 10647

	baseline	**NeLex**
avg	0.4792	**0.5950**
nonZeroAvg	0.5142	**0.8692**

Tabela T12 – wyniki tłumaczenia polsko-angielskiego

Komentarz

System *baseline* otrzymał dość wysokie oceny tłumaczenia, co jest zasługą odpowiedniego odnalezienia przykładów w pamięci tłumaczeń (przez system MAT). Nie mniej jednak, zauważalny jest bardzo wyraźny wzrost jakości tłumaczenia dla systemu NeLex, zwłaszcza biorąc pod uwagę parametr **nonZeroAvg**.

7.1.3. Tłumaczenie polsko-niemieckie

Wyniki dla tłumaczenia z języka polskiego na język niemiecki:

Liczba zdań do przetłumaczenia (moc zbioru T) : 7602

	Baseline	**NeLex**
avg	0.3639	**0.5656**
nonZeroAvg	0.3745	**0.8338**

Tabela T12 – wyniki tłumaczenia polsko-niemieckiego

Komentarz:

Ponownie można zaobserwować duży wzrost jakości tłumaczenia podczas stosowania systemu NeLex. Wyniki są nieznacznie gorsze od wyników dla tłumaczenia polsko-angielskiego, ze względu na to, że system nie dysponuje słownikiem do tłumaczenia polsko-niemieckiego.

7.1.4. Tłumaczenie polsko-hiszpańskie

Wyniki dla tłumaczenia z języka polskiego na język hiszpański:

Liczba zdań do przetłumaczenia (moc zbioru T) : 7499

	Baseline	**NeLex**
avg	0.3890	**0.4610**
nonZeroAvg	0.3981	**0.7476**

Tabela T13 – wyniki tłumaczenia polsko-hiszpańskiego

Komentarz:

Raz jeszcze można zaobserwować duży wzrost jakości tłumaczenia podczas stosowania systemu NeLex w stosunku do *baseline*. Wyniki ponownie są nieznacznie gorsze w stosunku do wyników tłumaczenia polsko-angielskiego, ze względu na brak słownika do tłumaczenia polsko-hiszpańskiego.

7.2. TEST FUNKCJI OBLICZAJĄCEJ OCENĘ TŁUMACZENIA

7.2.1. Przygotowanie testu

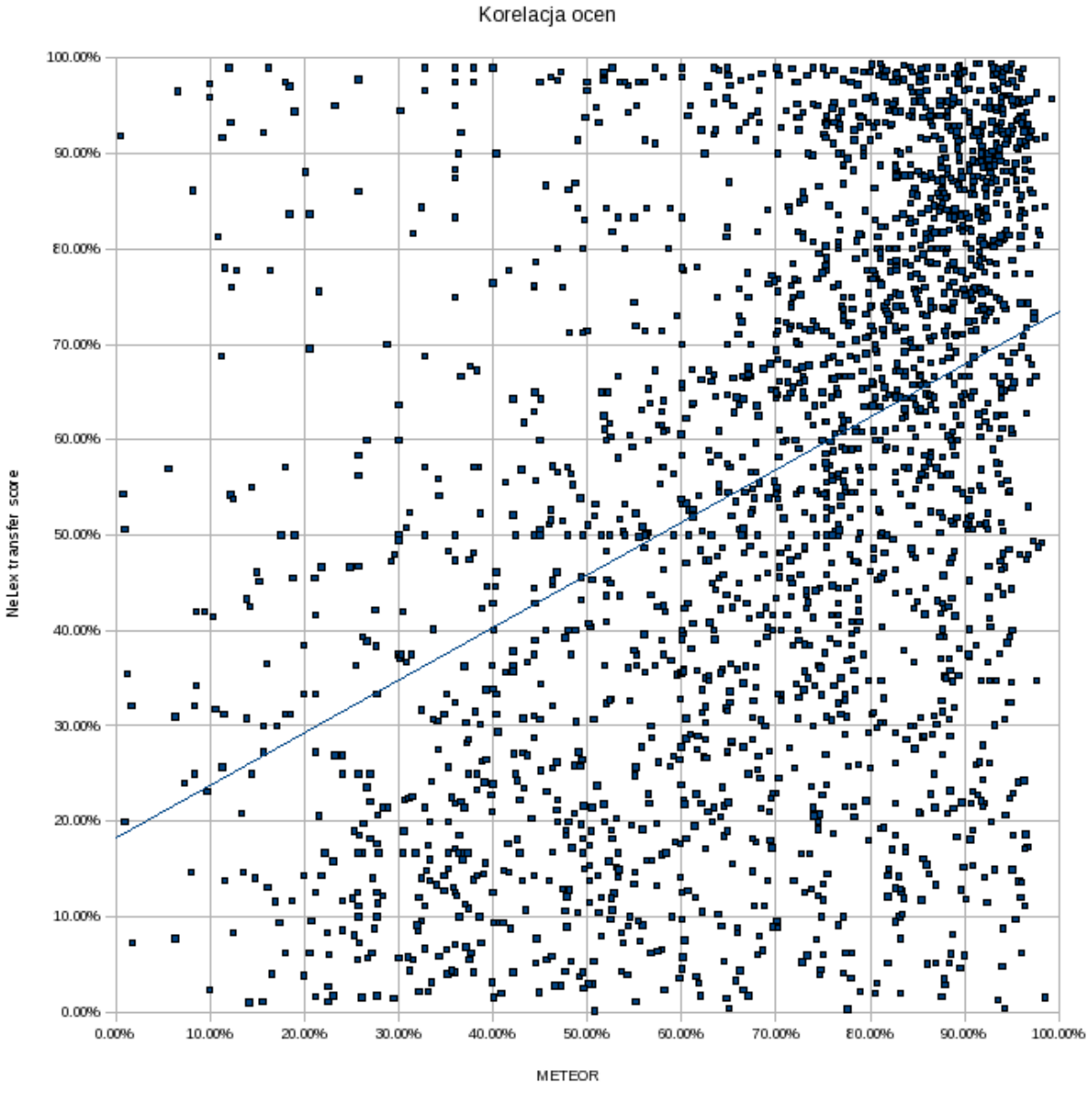
Jak już zostało wspomniane, system NeLex po przetłumaczeniu zdania podaje własną ocenę tłumaczenia. Ocena ta ma dawać wyobrażenie o tym, jakiej jakości jest wykonane przez system tłumaczenie zdania. Funkcja obliczania oceny tłumaczenia została poddana testowi.

Test został wykonany podczas przeprowadzania tłumaczenia polsko-angielskiego, opisanego w paragrafie 7.1.2. Polegał na porównaniu ocen tłumaczenia, jakie podawała metryka METEOR dla poszczególnych tłumaczeń zdań z ocenami, jakie podawał system NeLex dla tłumaczeń tych zdań. Pozwoliło to na określenie ko-

relacji oceny metryki METEOR oraz oceny systemu NeLex, traktując je jako zmienne losowe o ciągłym rozkładzie.

Z testu wyłączone zostały takie tłumaczenia zdań, dla których zarówno metryka METEOR, jak i system NeLex podawały oceny 0% oraz te zdania, dla których obie oceny wynosiły 100%. Łącznie zdań takich było ok. 75% i w znaczący sposób wpływały na wartość współczynnika korelacji. Jednocześnie, z punktu widzenia obliczania oceny tłumaczenia zdania, przypadki 0% i 100% są przypadkami szczególnymi, dość mało interesującymi.

7.2.2. Wynik testu



Rysunek R10 – korelacja ocen tłumaczenia

Powyższy wykres obrazuje korelację oceny tłumaczenia zdania przez metrykę METEOR oraz przez system NeLex. Na wykresie zaznaczona jest linia regresji. Obliczona wartość współczynnika korelacji: **0.41**

Komentarz:

Przebieg linii regresji oraz wartość współczynnika korelacji wykazują, że oceny tłumaczenia, wykonywane przez system NeLex, są na ogół mniejsze od tych, wykonywanych przy pomocy metryki METEOR. System NeLex ocenia więc siebie bardziej surowo, niż metryka METEOR. Niemniej jednak, dodatni współczynnik korelacji i to o wartości bliskiej 0.5 wskazuje na pewne powiązanie oceny podawanej przez metrykę METEOR oraz przez system NeLex.

7.3. TEST SZYBKOŚCI TŁUMACZENIA

7.3.1. Przygotowanie testu

W tłumaczeniu przez analogie szczególnie ważna jest szybkość tłumaczenia. Dlatego jednym z wymagań, stawianych przed systemem NeLex w fazie jego planowania, była wydajność.

Test szybkości został skonstruowany tak, aby testować wyłącznie szybkość systemu NeLex. Gdyby testować szybkość tłumaczenia dokumentów przez analogie, wynik nie dałby żadnej informacji o szybkości działania systemu NeLex. Jest to spowodowane faktem, że podczas tłumaczenia dokumentu przez analogie, znaczącą część czasu zajmuje wyszukiwanie przykładów w pamięci tłumaczeń.

Test szybkości systemu NeLex przebiegał w następujących krokach:

1. Losowy wybór ok. 28000 par zdań **(s,t)** z korpusu dwujęzycznego JRC polsko-angielskiego i umieszczenie ich w zbiorze **T**
2. Dla każdej pary **(s,t)** należącej do **T** wykonanie tłumaczenia systemem NeLex dla: **srcTemplate = s, trgTemplate = t, srcInstance = w**, gdzie **w** jest losowo wybranym zdaniem takim, że **(w, z)** należy do **T** dla jakiegoś **z**.

Choć wyniki takich tłumaczeń na ogół nie miały sensu, to w każdym z nich system NeLex musiał zastosować większość lub nawet wszystkie kroki swojego algorytmu tłumaczącego.

Testy szybkości były wykonywane na komputerze wyposażonym w procesor Intel Pentium Dual Core E2220 2.40 GHz, posiadającym 2 GB pamięci RAM.

7.3.2. Wynik testu szybkości

Wyniki trzech pomiarów szybkości tłumaczenia systemu NeLex (ilość tłumaczeń: 28707):

Lp.	Czas trwania [s]	Średnia szybkość tłumaczenia [ilość zdań/s]	Średni czas tłumaczenia zdania [s]
1.	157.244	182.563	0.0055
2.	157.143	182.681	0.0055
3.	156.099	183.903	0.0054

Tabela T14 – wyniki testu szybkości systemu NeLex

Komentarz:

Taka prędkość tłumaczenia jest sukcesem w kontekście złożoności algorytmów systemu NeLex. Dzięki prędkości tłumaczenia rzędu 180 zdań na sekundę system NeLex może z powodzeniem być wykorzystywany w systemie heTMan. Jest bowiem szybszy od wykorzystywanego dotychczas systemu ANTRA (opisanego w paragrafie 4.2.2), którego prędkość tłumaczenia oscyluje w granicach 50 zdań na sekundę.

7.4. DYSKUSJA NAD WYNIKAMI PRACY I PERSPEKTYWY ROZWOJU IDEI

Ogólnie, wyniki uzyskane przez system NeLex napawają optymizmem. Stanowią motywację do dalszych prac nad systemem.

Należy jednak zwrócić uwagę na fakt, iż sposób testowania mógł nie być w stanie dobrze zmierzyć rzeczywistej przydatności systemu do tłumaczenia. Dzieje się tak między innymi dlatego, że metryka METEOR, ze względu na swoją specyfikę, nie wykazuje błędów stylistycznych w tłumaczeniach. A system NeLex, poprzez podstawianie słów, potrafi wprowadzić do tłumaczeń błędy stylistyczne lub nawet takie, które zmieniają sens zdania (takie jak opisane w paragrafie 1.3.3).

Z tego powodu, najważniejszym celem w przyszłych pracach nad systemem będzie udoskonalenie mechanizmu podstawiania słów. Być może wartościowym usprawnieniem byłoby wprowadzenie mechanizmu oznaczania części mowy wyrazów w przetwarzanych zdaniach. Dzięki temu system mógłby dokonywać tylko wybranych podstawień, na przykład „przymiotnik za przymiotnik" lub „rzeczownik za rzeczownik". Poza tym, podczas tłumaczenia słowa (słowo przeniesione ze zdania wejściowego do docelowego musi być przetłumaczone na język docelowy) system korzystałby z informacji o części mowy wyrazu i podawał odpowiednią słowoformę wyrazu przetłumaczonego na język docelowy.

System NeLex dysponuje obecnie jedynie słownikiem do tłumaczenia polsko-angielskiego. W ramach prac nad rozwojem systemu mogłyby zostać opracowane również inne słowniki. W pierwszej kolejności byłyby to słowniki języka niemieckiego oraz rosyjskiego, gdyż te właśnie języki są obecnie obsługiwane przez system Translatica ([Translatica]). Następnie mogłyby zostać opracowane słowniki języków romańskich – hiszpańskiego (system NeLex jest już wyposażony w reguły oznaczania jednostek nazwanych w tym języku) oraz francuskiego.

Jeszcze inną drogą rozwoju systemu NeLex może być rozwijanie mechanizmów oznaczania jednostek nazwanych. Mogłyby zostać opracowywane zestawy reguł dedykowanych do oznaczania jednostek nazwanych w tekstach specjalistycznych z innych dziedzin, niż prawo. Mogłyby powstać wersje systemu NeLex do tłumaczenia tekstów medycznych, informatycznych, reklamowych i wielu innych.

PODSUMOWANIE

W niniejszej pracy została zebrana dotychczasowa wiedza na temat tłumaczenia przez analogie. Zostały zaprezentowane dotychczasowe osiągnięcia w tej dziedzinie. Technika ta została również skonfrontowana z techniką tłumaczenia przez transfer.

Zostały również zebrane informacje na temat zagadnień lingwistyki komputerowej blisko związanych z tłumaczeniem przez analogie, takich jak: dzielenie tekstu na zdania, dopasowywanie tekstów oraz wyszukiwanie przykładów w pamięci tłumaczeń. Każde z tych zagadnień mogło być tematem znacznie dłuższego opracowania. W tej pracy zebrano jednak tylko najistotniejsze informacje związane z wymienionymi zagadnieniami.

W ramach pracy został zaimplementowany system NeLex – całkowicie nowy system tłumaczenia przez analogie. Jest on wykorzystywany przez komercyjny system o nazwie heTMan. Tym samym, system NeLex jest bardzo intensywnie testowany.

W pracy opisany jest szczegółowo algorytm tłumaczenia, wykorzystywany przez system NeLex. Ma on być odpowiedzią na niedoskonałości wcześniejszych rozwiązań. Kluczowym rozwiązaniem jest rozpoznawanie i tłumaczenie jednostek nazwanych przy tłumaczeniu przez analogie. Mechanizm ten jest stosunkowo prosty, ale w znaczący sposób wpływa na polepszenie jakości tłumaczenia. Algorytm uzupełnia tłumaczenie pojedynczych słów z wykorzystaniem słownika.

System został zrealizowany w taki sposób, aby łatwo można było go przystosować do obsługi kolejnych języków tłumaczenia. Dodanie obsługi kolejnego języka tłumaczenia sprowadza się do opracowania nowego słownika i/lub reguł oznaczania jednostek nazwanych. Żadna z tych czynności nie wymaga ingerencji w kod systemu NeLex.

BIBLIOGRAFIA

- [CAS, 2004] : “An EBMT System Based on Word Alignment”; HOU Hongxu, DENG Dan, ZOU Gang, YU Hongkui, LIU Yang, XIONG Deyi oraz LIU Qun; http://www.mt-archive.info/IWSLT-2004-Hou.pdf
- [Gale, Church, 1991] : “A Program for Aligning Sentences in Bilingual Corpora” William A. Gale, Kenneth W. Church, 1991, Meeting of the Association for Computational Linguistics.
- [Gintrowicz, 2007] : “Tłumaczenie automatyczne oparte na przykładach i jego rozszerzenia”; Jacek Gintrowicz, 2007; praca magisterska, napisana pod kierunkiem dra Krzysztofa Jassema
- [Globalizacja] : “Strona główna systemu Globalizator”, http://globalizator.pl
- [hunalign] : “hunalign – sentence level aligner”, http://mokk.bme.hu/resources/hunalign
- [Interlingua, 1995-2006] : “The Lexical Semantics of a Machine Translation Interlingua”; Rick Morneau; 1995-2006; http://www.eskimo.com/~ram/lexical_semantics.html
- [JRC] : “Language Technology at the European Commission - Joint Research Centre, Ispra - JRC-Acquis”; http://langtech.jrc.it/JRC-Acquis.html
- [Knight, 1999] : “A Statistical MT Tutorial Workbook”; Kevin Knight; 1999; Nieopublikowane.
- [LingPipe]: LingPipe Home page; Alias-i; 2003-2008; http://alias-i.com/lingpipe/
- [Lipski, 2007] : “Urównoleglanie tekstów dwujęzycznych na poziomie zdania”; Jarosław Lipski; 2007; praca magisterska, napisana pod kierunkiem dra Krzysztofa Jassema
- [MAT] : “MAT – biblioteka programowa rozmytej pamięci tłumaczeń”; M. Wypych; 2007; PolEng,
- [METEOR] : ”The METEOR Automatic Machine Translation Evaluation System”; http://www.cs.cmu.edu/~alavie/METEOR/
- [Mikrokosmos, 1996] : “Semantic Analysis in The Mikrokosmos Machine Translation Project”; Stephen Beale, Sergei Nirenburg, Kavi Mahesh; 1996; http://ilit.umbc.edu/SergeiPub/SemantAnalysis.pdf

- [Nagao, 1984] : „A framework of a mechanical translation between japanese and english by analogy principle”; Makoto Nagao; 1984; http://www.mt-archive.info/Nagao-1984.pdf
- [NERT, 2008]: “Semi-supervised Learning Rule Acquisition for Named Entity Recognition and Translation”; K. Jassem, M. Marcińczuk; 2008;
- [TMX] : “Translation Memory eXchange (TMX)” – specyfikacja formatu TMX, http://www.lisa.org/tmx
- [SRX] : “Segmentation rules eXchange (SRX)” – specyfikacja formatu SRX, http://www.lisa.org/Segmentation-Rules-e.40.0.html
- [Translatica] : “Strona główna systemu Translatica”; http://www.translatica.pl
- [Translatica Server] : Translatica Server – opis produktu na stronie producenta – firmy POLENG Sp. z o.o.; http://www.poleng.pl/poleng/node/37