Biuletyn WAT
Vol. LXVII, Nr 2, 2018

Zastosowanie algorytmów normalizacji tekstu na potrzeby syntezy mowy w urządzeniach przenośnych

ARTUR ZACNIEWSKI[1], MARCIN KLEINSZMIDT[2]

[1]Akademia Marynarki Wojennej w Gdyni, Wydział Nawigacji **i** Uzbrojenia Okrętowego,
ul. Śmidowicza 69, 81-103 Gdynia, a.zacniewski@amw.gdynia.pl
[2]Toucan Systems, Sp. z o.o., Al. Grunwaldzka 103, 80-244, Gdańsk,
m.kleinszmidt@toucan-systems.pl

Streszczenie. W artykule pokazano kolejne etapy występujące w syntezie mowy, a także sposoby postępowania z poszczególnymi fragmentami tekstu, który ma zostać przetworzony na mowę. Przedstawiono wyniki badań wydajności algorytmów normalizacji treści realizowanych na potrzeby projektu Toucan Eye — urządzenia przenośnego z systemem sztucznej inteligencji, mającego wspomóc osoby z dysfunkcją wzroku. Pokazano, jak istotne są dobranie i optymalizacja zastosowanych algorytmów ze strony implementacyjnej, po to by zwiększyć komfort użytkownika końcowego.
Słowa kluczowe: synteza mowy, wspomaganie osób z dysfunkcją wzroku, Toucan Eye
DOI: 10.5604/01.3001.0012.0957

Wstęp

W skład systemów sztucznej inteligencji wspomagających osoby z dysfunkcją wzroku bardzo często wchodzi system syntezy mowy. Na wejście takiego systemu podawany jest tekst, np. z systemu przetwarzania obrazów, który wykrywa, przetwarza i rozpoznaje tekst znajdujący się w zasięgu użytkownika urządzenia.

Synteza mowy jest wieloetapowym procesem, którego efektem końcowym jest wytworzenie sygnału audio zawierającego tekst jak najbardziej przypominający efekt, jaki dałoby przeczytanie tego tekstu przez człowieka. W procesie syntezy mowy wyróżnić można m.in. następujące etapy:

— przetwarzanie tekstu naturalnego NLP (ang. *Natural Language Processing*), który zawiera szereg procesów wykonywanych po stronie tekstowej, przede

wszystkim normalizacja tekstu, ale także zwrócenie uwagi na niuanse kontekstowe, które mogą zmieniać pożądany sposób wymówienia danego słowa (dotyczy to zwłaszcza liczb, gdzie skomplikowane reguły koniugacyjne czynią to zadanie stosunkowo złożonym). Wyróżnić można tu m.in. fonetyzację i generację prozodii [2];

— etap syntezy właściwej, w której algorytm pracujący najczęściej na odpowiednio przygotowanym zestawie próbek zajmuje się przetwarzaniem spreparowanego już tekstu na sygnał audio [7].

Gdzieś pomiędzy nimi istnieje jeszcze subtelna warstwa intonacji i akcentowania, bardzo czytelna dla człowieka, choć niebezpośrednio rzutująca na poziom zrozumienia samego tekstu, a więc komunikacja — wciąż werbalna, ale nieprzywiązana ściśle do tekstu — niosąca dodatkowe informacje o stanie emocjonalnym osoby czytającej tekst w trakcie zdania [1].

Zmiany szybkości czytania, melodii głosu czy akcentowanie poszczególnych wyrazów czy głosek mogą sugerować skupienie uwagi na konkretnych aspektach wypowiadanej frazy. Jest to jednak zagadnienie bardzo złożone — nawet najlepsze na świecie syntezatory mowy w pewnych sytuacjach potrafią zabrzmieć nienaturalnie. Efekty te mogą świadczyć najwyżej o pewnym wzroście komfortu czy naturalności obcowania z systemem, ale nie wpływają w znacznym stopniu na skuteczność systemu jako pomocy dla osób z upośledzeniem wzroku.

1. Algorytmy normalizacji tekstu

Normalizacja tekstu ma za zadanie przetworzenie tekstu i nadanie mu spójnej formy ułatwiającej dalszą interpretację. Założeniem normalizacji jest zmiana formy przetwarzanego tekstu z formy pisanej na mówioną. Ze względu na złożoność i różnorodność języka polskiego normalizacja treści nie jest zadaniem trywialnym, składa się z dużej liczby złożonych etapów, a w wielu przypadkach wymaga stworzenia nowych mechanizmów przetwarzających tekst w sposób, który będzie pomocny dla przyszłych użytkowników.

Przygotowanie tekstu, który następnie zostanie przekazany do syntezy mowy, nie jest prostym zagadnieniem i od jakości tego przygotowania w dużej mierze zależy jakość i zrozumiałość mowy. Podejście do przygotowania treści będzie odmienne dla różnych języków i dialektów, gdyż każdy ma inną składnię i reguły językowe [3].

Etapy obejmujące zadanie normalizacji tekstu to:

— tokenizacja, czyli przekształcenie tekstów w usystematyzowany zbiór „tokenów" (znaczników z polami pomocnymi przy dalszej obróbce i analizie). Pomaga algorytmowi w „zrozumieniu" poszczególnych części tekstu, determinuje, w jaki sposób ma być interpretowane dane słowo, biorąc pod uwagę kontekst oraz otaczającą treść;

— odrzucenie słów i znaków nieistotnych lub zamiana ich na znaki równoważne (np. dla syntezy mowy wymowa „ó" będzie taka sama jak „u");
— zamiana wyrazów z wybranych grup tokenów na reprezentację słowną danego wyrażenia. Odrębnych mechanizmów translacji (m.in. tworzenia słowników) wymagają liczby naturalne, liczby rzeczywiste, liczby poprzedzone lub zakończone symbolami, daty, godziny, symbole i znaki, skróty i skrótowce, adresy email, strony www, znaki tabulacji lub końca linii [4].

Faza ta obejmuje czynności, których człowiek posługujący się biegle językiem nie dostrzega i nie zdaje sobie sprawy ze stopnia ich trudności. Przykładowo problem można dostrzec u obcokrajowców próbujących nauczyć się języka polskiego, mają oni spore trudności z dobraniem odpowiedniej formy wypowiedzi i takie same problemy napotykają twórcy algorytmów przygotowujących tekst do syntezy mowy [2].

1.1. Tokenizacja

W kolejnych etapach działania algorytmu badania kontekstu tokeny mogą być uzupełniane kolejnymi informacjami potrzebnymi z punktu widzenia syntezy mowy. Przykładem takim może być klasyfikacja znaków, np. przecinka lub dwukropka, znaki takie określają sposób czytania następujących po nich treści.

Token może być reprezentowany w standardzie XML (ang. *Extensible Markup Language*), rozwiązanie takie jest proste i efektywne, w pełni wystarczające dla potrzeb projektu. Każdy token zawiera pole „id", dzięki czemu algorytm może sprawdzić wcześniejsze i kolejne słowa podczas analizy kontekstu lub odtworzyć tekst wejściowy. Kolejnym polem jest „word", gdzie przechowywana jest oryginalna wartość słowa. Pole „type" określa typ słowa. Determinuje on metody późniejszego przetwarzania danych słów. Pole „value" zawiera tłumaczenie znaków niebędących literami na odpowiednią interpretację słowną. Wartość tego pola będzie dalej modyfikowana i w ostatniej fazie tłumaczona na słowniki fonetyczne [5].

Struktura tokenu ma następującą budowę:

```
<te id="ID_słowa_w_treści" word="słowo" type="typ_słowa" value="" />    (1)
```

Przykład tokenizacji dla tekstu „Sklep otwarty od 08:30":

```
<te id="1" word="Sklep" type="wordFirstUppercase" value="sklep" />
<te id="2" word="otwarty" type="wordLowercase" value="otwarty" />
<te id="3" word="od" type="wordLowercase" value="od" />
<te id="4" word="08:30" type="hour" value="ósma trzydzieści" />
```

Tokenizacja tekstu pozwala na wskazanie w metodzie translacyjnej, w jaki sposób dane słowo ma zostać odczytane. Umiejętność dobrania odpowiedniej metody i sposobu translacji jest podstawowym zagadnieniem efektywnej syntezy mowy. Oczyszczanie tokenów ma na celu usunięcie pól niemających znaczenia dla

dalszych etapów syntezy mowy, co pozwala na przyspieszenie działania algorytmu oraz poprawę jakości syntezowanej mowy. Na potrzeby badań opracowano algorytmy normalizacji dla wszystkich ww. grup, ale ze względu na objętość artykułu pokazano tylko wybrane [6].

2. Algorytm normalizacji tekstu — daty

Przebieg pracy algorytmu dla rozpoznanej daty niezawierającej spacji:

- Na początku algorytm wykonuje detekcję znaków rozdzielających datę, mogą nimi być znaki: '/', '-', lub '.';
- W następnym kroku ustalana jest pozycja dnia oraz roku w dacie, przy założeniu pozycji miesiąca pośrodku daty, tak jak ma to miejsce w ustalonym formacie daty dla Polski;
- Następnie data oraz rok są zamieniane z formy liczbowej na pisaną przy użyciu algorytmu do zamiany liczb naturalnych. Dodatkowo algorytm do zamiany roku na wejściu otrzymuje informację, aby korzystać z innego słownika zawierającego formy specjalnie przygotowane dla niego;
- W ostatnim kroku miesiąc jest zamieniany zgodnie ze słownikiem Miesiące, gdzie ewentualne zero przed liczbą jest usuwane.

Metoda ta jest stosowana dla tokenów posiadających typ 'date'. Złożoność tej metody podkreśla fakt, że data może być zapisywana na wiele sposobów. Dodatkowo odczytanie może być różne dla odmiennych przypadków. Algorytm będzie oczekiwał trzech wartości:

— dzień miesiąca w formacie 1-31, w tym dla dat 1-9 może być poprzedzone 0,
— numer miesiąca w postaci liczb arabskich, rzymskich lub w postaci skrótu oraz pełnej nazwy miesiąca,
— rok w postaci YYYY lub YY.

Wartości te mogą występować w różnej kolejności i mogą być oddzielane znakami:

— spacją (DD MM YYYY),
— myślnikami (DD-MM-YYYY),
— kropkami (DD.MM.YYYY),
— ukośnikami (DD/MM/YYYY),
— mieszane (DD MM, YYYY).

Skuteczność (stosunek poprawnie rozpoznanych dat do liczby wszystkich badanych dat) opisywanej metody dla składowych rozdzielonych znakiem '-' lub '.' (kropką) wyniosła w badaniach 97,5%.

Dla normalizacji godzin (typ 'hour') uzyskano skuteczność około 95%. Błędy wynikały ze złego doboru formy wypowiedzi dla niektórych przypadków.

3. Algorytm normalizacji tekstu — skróty i skrótowce

Ta metoda jest używana dla tokenów typu 'shortcut' i powinna być rozpatrywana względem wartości liczby mnogiej lub pojedynczej. Do realizacji opisanej metody konieczne jest zdefiniowanie dwóch zbiorów. Algorytm tworzy na ich podstawie słowniki. Jeden dla skrótów zakończonych kropkami, np.: prof., inż., p.n.e. oraz drugi dla skrótów bez kropek, np.: PLN, zł, ABW, RP. Jeśli badane słowo pasuje do wzorca dla skrótów z kropką, algorytm przeszukuje odpowiedni słownik oraz jeśli występuje w nim klucz odpowiadający skrótowi, zwraca jego rozwiniętą formę. W przeciwnym razie, jeśli badane słowo nie pasuje do wzorca, tzn. nie zawiera kropki, algorytm przeszukuje słownik, w którym znajdują się skróty bez kropek. Jeśli znajdzie odpowiedni klucz, zwraca jego wartość. Obecnie lista skrótów posiada około 200 pozycji. Skuteczność opisywanego algorytmu wynosi 90%.

W przypadku normalizacji liczb naturalnych (typ 'numberNatural') dla 500 losowo wygenerowanych liczb z zakresu 1-9 999 999 999 999 999 999 uzyskano skuteczność 100%.

4. Algorytmy normalizacji tekstu — liczby rzymskie

Algorytm polega na zamianie liczb rzymskich na arabskie, następnie wynik operacji przekazywany jest do opisanej już metody normalizacji liczb arabskich. W systemie rzymskim liczby zapisywane są od lewej do prawej, zaczynając od liczb od największej wartości. Istnieją wyjątki od tej reguły dla liczby 4 oraz 9 i ich krotności dziesiętnych.

Algorytm został przebadany na grupie 520 losowo wygenerowanych liczb w formacie rzymskim w zakresie 1-2050. Skuteczność opisywanego algorytmu to ok. **99,9%**. Jedyny błąd algorytmu to zakwalifikowanie liczby 'CV' jako skrótu i odczytanie jej jako 'curriculum vitae'.

5. Algorytmy transkrypcji fonetycznej

Tekst, który przeszedł już przez wszystkie etapy normalizacji, nie nadaje się jeszcze do syntezy, gdyż sam tekst w postaci słownej nie niesie żadnych informacji o sposobie wypowiedzi. Nie pozwala on na jednoznaczne określenie wymowy danego wyrazu, dobrym przykładem obrazującym problem może być słowo auto, gdzie „u" musi zostać przeczytane jako „ł" [9].

Do zapisu wymowy może służyć alfabet fonetyczny IPA (ang. *International Phonetic Alphabet*) lub SAMPA (ang. *Speech Assessment Methods Phonetic Alphabet*). Translacja polega na zamianie liter danych wejściowych na odpowiadające im

symbole w danym alfabecie fonetycznym. Należy także wyróżnić reguły precyzujące translację wyjątków dla języka polskiego. W tabeli nr 1 pokazano translację znaków wykorzystywaną podczas badań. Alfabet SAMPA zapisywany jest w kodzie ASCII, dzięki czemu korzystanie z niego jest bardziej praktyczne dla deweloperów [8].

Tabela 1

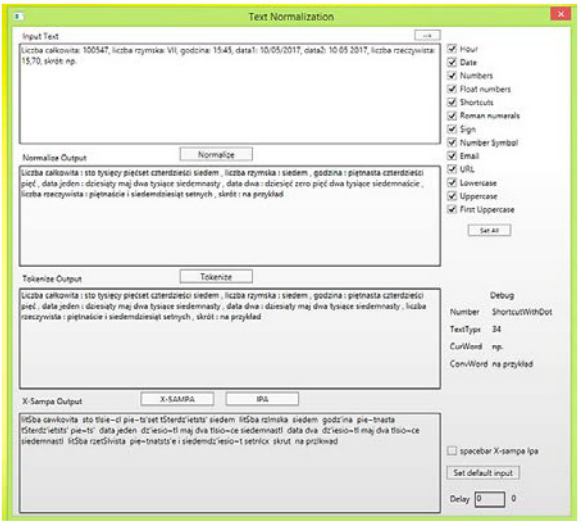
Translacja znaków w alfabecie SAMPA

Symbol	SAMPA	Symbol	SAMPA	Symbol	SAMPA
a	a	k	k	w	v
ą	o~	l	l	y	I
b	b	ł	w	z	z
c	c	m	m	ź	z’
ć	ts’	n	n	ż	Z
d	d	ń	n’	sz	S
e	e	o	o	ś	ts
ę	e~	p	p	dz	dz
f	f	r	r	cz	tS
g	g	s	s	dż	dZ
h	x	t	t	dź	dz’
i	i	u	u	rz	Z
j	j	ó	u		

6. Wydajność metod normalizacji treści

Szybkość działania metod normalizacji tekstu ma kluczowy wpływ na działanie całej syntezy, gdyż zbyt długie przetwarzanie tekstu wydłuży czas oczekiwania na dźwięk, co w konsekwencji może prowadzić do tego, że urządzenie będzie czytało napisy, które już nie są istotne dla użytkownika. Testy zostały przeprowadzone na komputerze z procesorem Intel Core i5-4460 oraz 8 GB pamięci RAM. Programy testowe zaimplementowano w języku C# wraz z graficznym interfejsem użytkownika (GUI) pokazanym na rysunku 1.

Do sprawdzenia wydajności przygotowano 10 zdań o różnej długości, zawierających elementy wymagające translacji lub usunięcia (liczby arabskie, rzymskie, daty, godziny, skróty oraz znaki interpunkcyjne). Czasy wykonywania poszczególnych operacji przedstawia tabela nr 2, przy czym X-SAMPA to rozszerzony alfabet SAMPA.



Rys. 1. GUI programu do normalizacji treści [opracowanie własne]

Tabela 2

Wydajność w wybranych etapach przetwarzania tekstu

Liczba znaków w tekście	Czas normalizacji (ms)	Czas tokenizacji (ms)	Czas translacji X-SAMPA (ms)	Całkowity czas (ms)	Użycie CPU (%)	Użycie RAM (MB)
214	3,000	3,125	1,571	7,696	8	60
9	2,142	2,428	1,285	5,857	4	60
1124	8,875	6,875	3,500	19,250	15	60
1760	13,75	7,000	5,125	25,875	17	62
717	8,000	6,000	2,375	16,375	14	60
92	3,375	3,500	1,250	8,125	12	60
1786	14,125	8,500	5,500	28,125	20	62
1343	11,250	6,875	3,625	21,750	10	60
Średnia:	**8,064**	**5,537**	**3,029**	**14,784**	**12,5**	**60,5**

Dla każdej konkretnej liczby znaków w tekście (pierwsza kolumna) przeprowadzono 10 pomiarów, a w tabeli pokazano wartości średnie tych pomiarów. Wartość wariancji poszczególnych wyników nie była większa w żadnym przypadku niż 12% wartości średniej. Średnia czasu całkowitego to około 15 ms i biorąc pod uwagę, że w badaniu wykorzystano kilka długich fragmentów tekstu (1700 znaków — trzy akapity tekstu), możemy uznać wynik za zadowalający. Całkowity czas syntezy to

suma czasów normalizacji treści i syntezy mowy, inaczej mówiąc, czas od momentu podania tekstu na wejście systemu syntezy mowy do czasu rozpoczęcia odczytu przez urządzenie.

Podczas analizy wydajności algorytmu ważnym elementem jest również koszt obliczeniowy danej metody. W projekcie Toucan Eye, na potrzeby którego realizowane były ww. algorytmy i badania, ma to szczególne znaczenie, gdyż urządzenie powinno mieć kompaktowe rozmiary, a zasilanie będzie bateryjne. Średnie użycie procesora dla przykładów opisanych powyżej to 12,5%, a średnie wykorzystanie pamięci RAM to 60,5 MB. Dla urządzenia przenośnego zużycie zasobów najprawdopodobniej będzie większe, ale rząd wielkości pozostanie ten sam. W tej fazie projektu skupiono się na poprawności działania algorytmów, natomiast w kolejnych etapach algorytmy te będą optymalizowane pod kątem użycia w urządzeniu przenośnym.

7. Wnioski

Analizując wydajność mechanizmów i algorytmów normalizacji tekstu, można stwierdzić, że zarówno czas potrzebny na przetwarzanie treści, jak i wykorzystanie zasobów urządzenia jest na bardzo niskim poziomie i nie powinien mieć wpływu na poprawne działanie systemu. Dalsze praktyczne testy pozwolą na weryfikację osiągniętych rezultatów.

Badania dotyczą projektu „System sztucznej inteligencji wspomagający osoby z dysfunkcją wzroku — Toucan Eye" realizowanego w ramach Programu Operacyjnego Inteligentny Rozwój 2014-2020 działanie 1.1/poddziałanie 1.1.1 i finansowanego przez Narodowe Centrum Badań i Rozwoju.

Artykuł wpłynął do redakcji 3.11.2017 r. Zweryfikowaną wersję po recenzjach otrzymano 14.12.2017 r.

LITERATURA

[1] Delgado R., Araki M., Neto J., *Spoken, Multilingual and Multimodal Dialogue Systems: Development and Assesment*, Wiley, USA, 2005.

[2] Graliński J., Jassem K., Wagner A., Wypych M., *Linguistic aspects of text normalization in polish text-to-speech system*, System Science, vol. 32, no. 4, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2006, pp. 7-15.

[3] Łopatka K., Czyżewski A., *Syntetyzer mowy uwzględniający prozodię wypowiedzi*, Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej, nr 28, 2010, s. 105-108.

[4] NKJP, http://nkjp.pl/, strona Narodowego Korpusu Języka Polskiego, dostęp — wrzesień 2017.

[5] Perkins J., *Python 3 Text Processing with NLTK 3 Cookbook*, Packt Publishing, Birmingham, UK, 2014.

[6] Sołdacki P., *Zastosowanie metod płytkiej analizy tekstu do przetwarzania dokumentów w języku polskim*, rozprawa doktorska, Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych, Warszawa, 2006.

[7] PJWSTK, http://syntezamowy.pjwstk.edu.pl/synteza.html, strona Polsko-Japońskiej Szkoły Technik Komputerowych o syntezie mowy polskiej, dostęp — wrzesień 2017.

[8] SAMPA, http://www.phon.ucl.ac.uk/home/sampa/, strona alfabetu fonetycznego SAMPA, dostęp — wrzesień 2017.

[9] Tadeusiewicz R., *Sygnał mowy*, Wydawnictwa Komunikacji i Łączności, Warszawa, 1988.

A. ZACNIEWSKI, M. KLEINSZMIDT

Analysis of speech synthesis algorithms for the purposes of deployment in embedded device

Abstract. The article presents consecutive stages of speech synthesis and also the ways of dealing with particular fragments of a text. The results of performance measurement for the text content normalization algorithms are shown. These algorithms were developed for the Toucan Eye project – an embedded device with an artificial intelligence system able to help people with impaired sight. It was shown how essential is the choice and optimization of the applied algorithms for the implementation process in order to increase the end-user's comfort.

Keywords: speech synthesis, assisting persons with impaired sight, Toucan Eye
DOI: 10.5604/01.3001.0012.0957