




# **МОРФОЛОГИЯ НА АНГЛИЙСКИЯ ЕЗИК**



Морфологията е учение за начина, по който думите са изградени от по-малки пълнозначни единици, наречени морфеми.

Можем успешно да разделим света на морфемите в два широки класа:

- корени – сърцевината на пълнозначните частици в лексикона
- наставки – частици и парчета, които се комбинират с корените, за да преобразуват техните значения и граматични функции.

Възможно е също успешно да разделим морфологията на два големи класа: флексивна (синтетична) морфология и етимологична морфология.



## **Класове думи**

Под класове думи имаме предвид понятията като съществителни, прилагателни и глаголи

## Съществителни и глаголи (в Английския език)

Съществителните имат проста синтетична морфология, отличители за множествено число и отличители за притежание. Глаголите са по-сложни – отличителите са подходящи за функциите, на които глаголите се очаква да служат.

## Правилни и неправилни глаголи

Морфологични класове	Правилно спрегнати глаголи			
корен	walk	merge	try	map
-s форма	walks	merges	tries	maps
-ing причастие	walking	merging	trying	mapping
минала форма –ed	walked	merged	tried	mapped
причастие				

Морфологични класове	Неправилно спрегнати глаголи		
корен	eat	catch	cut
-s форма	eats	catches	cuts
-ing причастие	eating	catching	cutting
минала форма	ate	caught	cut
-ed причастие	eaten	caught	cut

## FSAs и двусмислието

В ND-FSAs разпознаването може да има голям брой пътища през четящо до прието състояние. Няма значение кой от тях е открит. В FSAs пътя, взет през машината всъщност има значение.

Например анализираме unionizable:

union-ize-able

un-ion-ize-able

Всяко представлява валиден път през машината, отразяващ се на различен анализатор



Има доста начини за управление на този проблем

- приемаме, че първата успешна структура пише на лентата
- прекарваме анализатор през всички възможни пътища, изписвайки всички валидни структури по пътя
- отклоняваме търсенето по същия начин, така че само един (или няколко) валидни пътеки да са всъщност търсените.

Повечето от тези ред действия включват начини за осъществяване на финалната опция.

Unionizable представлява проблем, включващ глобално двусмислие.

# Правопис

## Правила за правопис

Име	Описание на правилата	Пример
Удвояване на съгласни	1 съгласна буква се удвоява преди –ing/-ed	beg/begging
Зачертаване на е	Пропускане на е преди –ing/-ed	make/making
Вмъкване на е	Прибавяне на е след –s,-z,-x,-ch, -sh, преди -s	watch/watches
Местене на у	-у се променя в -ie преди –s, -i преди -ed	try/tries
Вмъкване на k	Глаголи, завършващи с гласна и –с се добавя k	panic/panicked



## Multi-Level и Multi-Tape машини

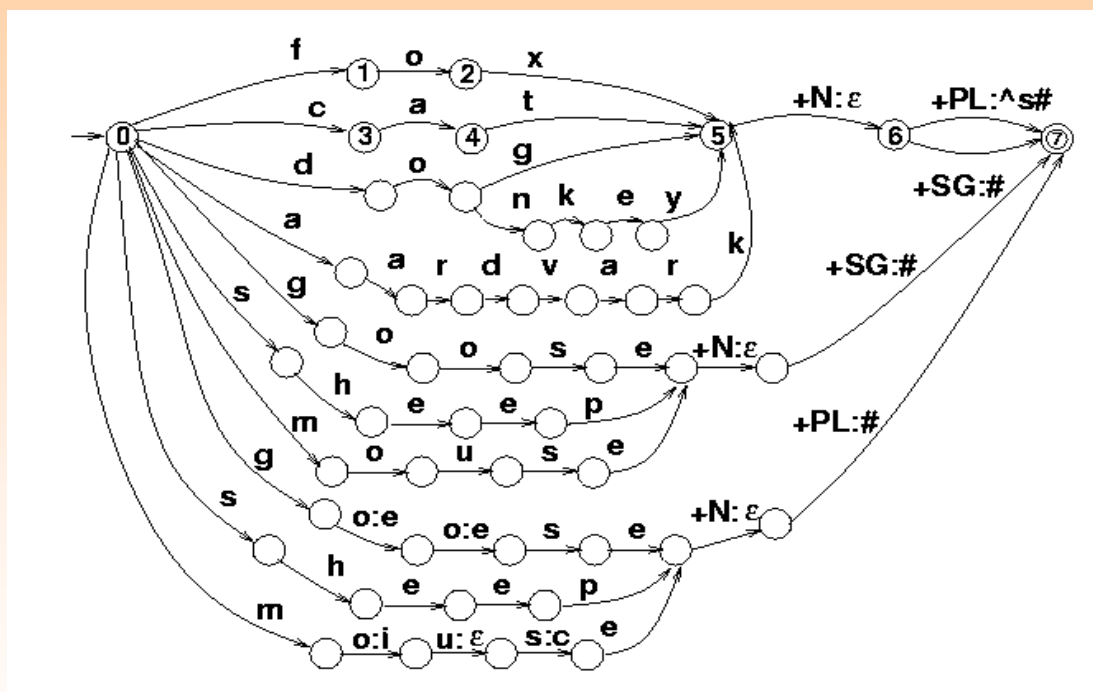
Често използваните FST идиоми, наричани каскади, са за да има изпадане на едно FST четене, както и да има вмъкване на последователни машини.

За да работим с правописа ние използваме три вида записи: лексикални, междинни, повърхностни.

<i>Lexical</i>	{	f	o	x	+N	+PL		}
<i>Intermediate</i>	{	f	o	x	^	s	#	}
<i>Surface</i>	{	f	o	x	e	s		}

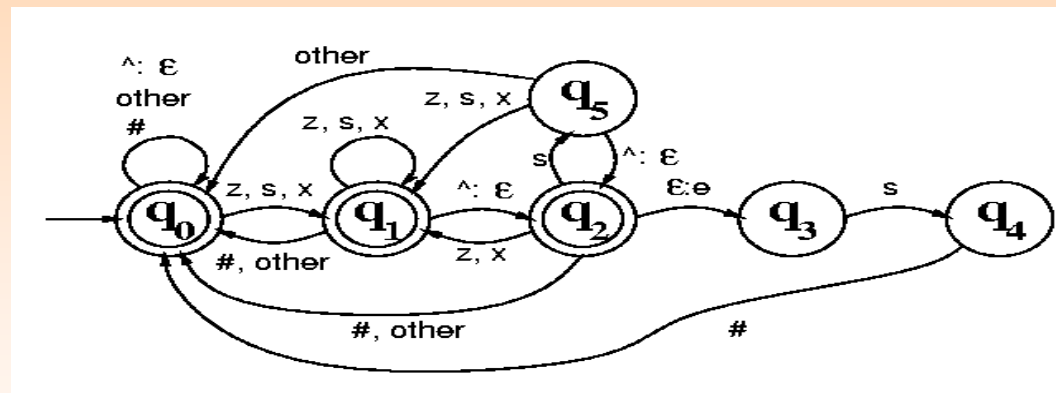
Нуждаем се от един преобразовател, за да работим между лексикалните и междинните нива и второ да работим между междинните и повърхностните нива, за да изгладим правописа.

## Лексикални и междинни преобразователи



## Междинни и повърхностни

Имаме нужда от FST, за да прибавим  $\epsilon$  между  $x$ ,  $s$  или  $z$  и пред  $s$ , в края на думата.



Ключовият момент на тези FST е, че са несвързани и минават невредими.

## Портър Стеминг

(m>0) ATIONAL> ATE

(m>0) TIONAL> TION

(m>0) ENCI> ENCE

(m>0) ANCI> ANCE

(m>0) IZER> IZE

(m>0) ABLI> ABLE

(m>0) ALLI> AL

(m>0) ENTLY> ENT

(m>0) ELI> E

(m>0) BILITI> BLE

relational> relate

conditional> condition

rational> rational

valenci> valence

hesitanci> hesitence

digitizer> digitize

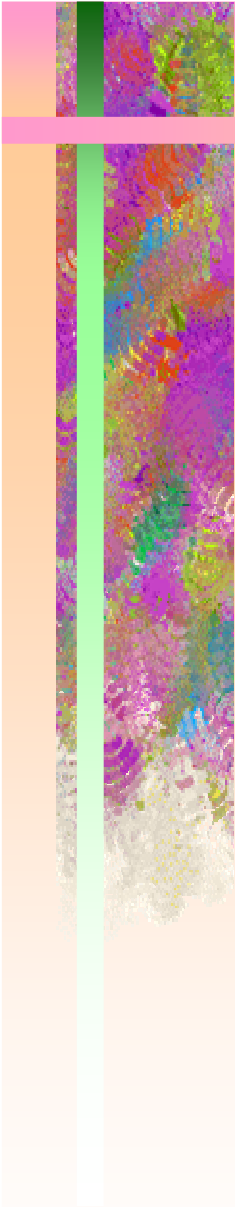
conformabli> conformable

radicalli> radical

diffetently> different

vileli> vile

sensibiliti> sensible




Портър Стеминг алгоритъмът е процес за отстраняване на по-общите морфологични и флексионни окончания за думите в Английския език.

Алгоритъмът първоначално е описан от Портър Стеминг, през 1980 г. Служи за премахване на наставки. След това е бил препечатан от Спрек Джонс, Корън и Питър Уилет през 1997 г. в Readings in Information Retrieval, Сан Франциско.

Алгоритъмът е широко използван, цитиран и преработван повече от 20 години. За нещастие, вариантите за него изобилстват, което изисква да се избере верният и това може да причини объркване.

Следващият пример е за алгоритъм на Стемер, написан на ANSI C:



Оригиналният алгоритъм е кодиран на BCPL, език, който вече не е популярен. В неговата крайна оцеляла форма, тази BCPL версия има три маловажни точки, за разлика от публикувания алгоритъм, и това несъмнено е отбелязано в ANSI C версията.

Алгоритмите на ANSI C, Java и Perl са еквивалентни на BCPL версията, които са тествани подробно с Английски текст. Оригиналният доклад е свързан с анулиране на типове, но даже и така да е, ANSI C версията действа като по-добро определение на алгоритъма, отколкото оригинално публикувания доклад.

## Разлики относно публикувания алгоритъм

-Има ново правило в Стъпка2,  $(m>0) \log y > \log$ . Така че, archeology е равностойно на archeological.

-Стъпка2, правило  $(m>0) abli > able$  е заместен с  $(m>0) bli > ble$ . Това е подобрене, possible е равностойно на possible.

Алгоритъмът не се отнася за низове с дължина 1 и 2. Във всеки случай низът с дължина 1 няма да се промени, ако премине през алгоритъма, но низ с дължина 2 може да изпусне последното s, така че коренът as евентуално да е a и is - i. Това едва ли има значение, тъй като в постановката на IR системата as, is, a и i вероятно са краища на думи, но във всеки случай няма да отстрани s.

Тази последна разлика, може би е винаги факт, когато се представя програмата за която се публикува подобен алгоритъм. Но след като са изминали 20 години от оригиналната публикация, това е трудно да се каже.

Трябва да се наблегне на това, че тези различия са много малки в сравнение с измененията в другите кодирания на алгоритъма.

## Общи грешки в другите кодирания

Алгоритъмът несъмнено обяснява това кога поставените правила на типовете (conditoin)  $S1 > S2$  са представени заедно, кое правило е приложено, кое е с най-дълга съчетаваща наставка  $S1$  за думата, дали правилото преуспява или пропада (например дали  $S1$  заменя  $S2$ ). Въпреки това, в болшинството от кодирания, правилата са просто приложени по ред, докато който и да е един от тях преуспява или списъкът пропада.

Това води до малки грешки, на различни места, например в Стъпка4 правилата  $(m>1)element > (m>1)ment > (m>1)ent$  отстранява последните  $ement$ ,  $ment$ ,  $ent$ .

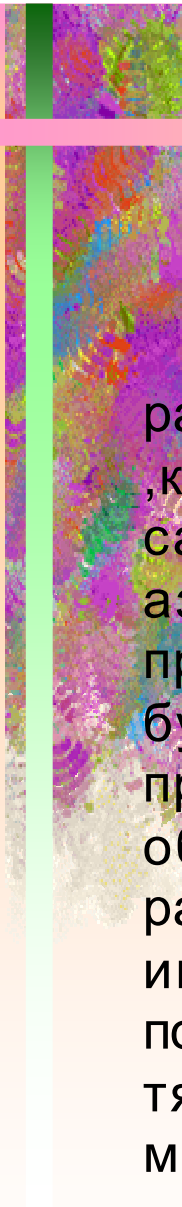


# Саундекс

## **Озвучаване и генеалогия (родословие) от Гари Мокотоф**

Кое може да бъде по обесърчаващо за генеалозите от това да погледнат записите по азбучен ред и да не могат да намерят това, което са търсили и после да го намерят по-късно, може би години по-късно, при положение, че информацията е била там през цялото това време, само че грешно произнесена(спелувана).Как ще намериш градове с имигрантски и не английско говорещ произход, когато единствената налична информация е преминала през генерацията устно и без значение както и да го произнасяш, не можеш да намериш града на картата.

За главното разрешаване на тези проблеми са се погрижили преди повече от 80 години когато Робърт С.Русел от Питсбург, Пенсилвания издава патент номер 1,261,167 на 2 април 1918 година за “изобретяването на ново сигурено и полезно подобрене в индексирането...както ще дава възможност на други квалифицирани в областта, към която се отнася , да правят и използват същото”.Идеята е за индексирането на информацията по скоро от гледна точка на това как тя звучи от колкото как азбучно се представя.Това станало известно като “soundexing”.



Русел записва във своя патент причината за разработването на тази система."Има сигурни звуци ,които формират ядрото на английския език и тези звуци са представени недопустимо само от буквите от азбуката, като един звук понякога може да бъде представен от повече от една буква или комбинация от букви, и една буква или комбинация от букви може да представя два или повече звука. Поради това, обикновено много имена могат да имат две или повече различни произношения, които в азбучен ред, или индексация , която отделя имената според последователността на буквите от азбуката в тях,изисква тяхното класиране в значително обособени места".

Той знаел, че буквите в азбуката били отделени, фонетично в категории. Всяка категория той обозначил с числова стойност. Както патента му описва:

1. Гласните (той ги нарича "устни резонанти") а, е, и, о, у, у.
2. Лабиалните или лабио-зъбни b, f, p, v.
3. Шептящите и съскащите c, g, k, q, s, z.
4. Зъбно-безгласните d, t.
5. Палатализираните l .
6. Самозараждащите се носови m .
7. Кънтящи или смесено-носови n .
8. Зъбно фрикативни r .



Има още няколко допълнителни правила:

- началната буква на думата винаги се запазва

- две последователни букви, които имат един и същи код се разглеждат като една буква(т.е. “tt” се кодира също като “t”)

- комбинацията “gh”, и “s”или”z”ако те са на края на думата се изоставят

- гласните (Group 1) се считат само ако се появят на първа позиция

## Американската Soundex система

Soundex кода съдържа първата буква на името следвана от три цифри. Тези три цифри се детерминират чрез изпускане на буквите а, е, і, о, u, h, w и у и добавяне на цифри от оставащите букви на името съобразно долната таблица. Тук има само две добавени правила (1) ако две последователни букви имат един и същи код, то те се кодират като една. (2) Ако има не достатъчно букви, за да се запишат три цифри, то оставащите се дописват с 0.

Soundex таблица

1 b, f, p, v

2 c, g, j, k, q, s, x, z

3 d, t

4 i

5 m, n

6 r



Примери:

Miller M460

Peterson P362

Peters P362

Auerbach A612

Uhrbach U612

Moskowitz M232

Moskovitz M213



## Детч-Мокотоф Soundex система

Последните значителни подобрения в системата е така наречената Daitch-Mokotoff Soundex система. През 1985 година нейния автор подредил имената на около 28,000 човека ,които легално променили имената си докато живеели в Палестина от 1921 година до 1948 година.Повечето от тях били евреи с германски или славянски фамилии. Очевидно било, че имало огромен брой варианти на произношение на еднакви основи на фамилиите и за това списъка трябвало да бъде подложен на тази система. Чрез използването на конвенционалната система на правителството на САЩ, базирана на системата на Русел, много източно европейски еврейски имена ,които звучали еднакво имали различен soundex код.Най-преобладаващи били тези имена произнасяни заменимо с буквите w или v, например, имената Moskowitz и Moskovitz.

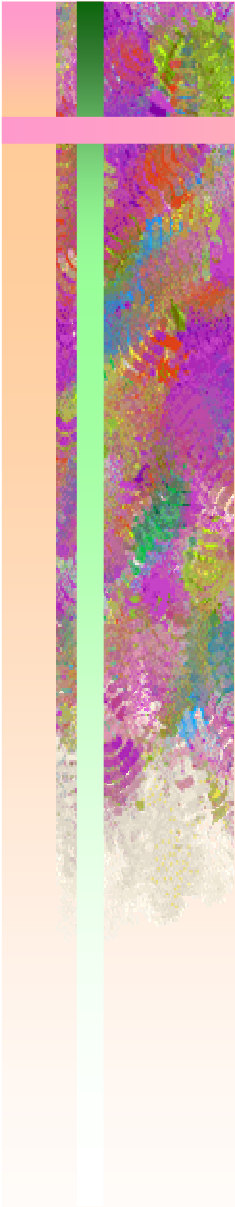
## Правила на Детч-Мокотоф звуковата система

1. Имената на градовете се кодират с шест цифри, както е представена от съответния звук от кодовата таблица отдолу.

2. Буквите А, Е, I, О, U, J и Y винаги се кодират в началото на думата, като в Augsburg (054795). В друга ситуация те се игнорират, освен когато две от тях сформират двойка, тя се намира преди гласна, както в Breuer (791900), но не както в Freud. Буквата "H" се кодира в началото на името както в Halberstadt (587943) или пък предхожда гласна както в Mannheim (665600), в други случаи не се кодира.

3. Когато близки звуци се комбинират да формират дълъг звук на тях им се дава кодово число на дълъг звук, както в Chernowitz, които не се кодират Chernowi-tz (496734), но Chernowi-tz (496740).

4. Когато близко стоящите букви имат еднакъв код, то те се кодират като един звук както Cherkassy, която не е кодирана Cherka-s-sy (495440), а Cherkassy (495400). Изключение правят буквените комбинации "MN" и "NM", които букви са кодирани различно както при Kleinman, която е кодирана 586660, а не 586600.



5. Когато име съдържа повече от една дума, то се кодира като една, като Nowy Targ, която се взима като Nowytarg.

6. Няколко букви или буквени комбинации поставят проблема, че те могат да прозвучат по два различни начина. Буквите и буквените комбинации CH, CK, C, J и RZ (виж таблицата отдолу) са означени с два възможни кода-номера. Бъди сигурен, че трябва да ги кодираш и по двата начина.

7. Когато името няма достатъчно кодирани звуци да запълни шесте цифри, останалите цифри се дописват с нули както при Berlin (798600), където има четири кодирани звука (B-E-R-L-I-N).

## Soundex за Българския език

Soundex кода съдържа първата буква на името следвана от три цифри. Тези три цифри се детерминират чрез изпускане на буквите а, е, и, о, ю, я, й, ъ, ъ и у и добавяне на цифри от оставащите букви на името съобразно долната таблица. Тук има само две добавени правила

(1) ако две последователни букви имат един и същи код, то те се кодират като една.

(2) Ако има не достатъчно букви да се запишат три цифри, то оставащите се дописват с 0.

Soundex таблица

1 б, ф, п, в, х

2 с, ц, з, ч, ш, щ, ж, г, к

3 д, т

4 л

5 м, н

6 р