



# Curs 12



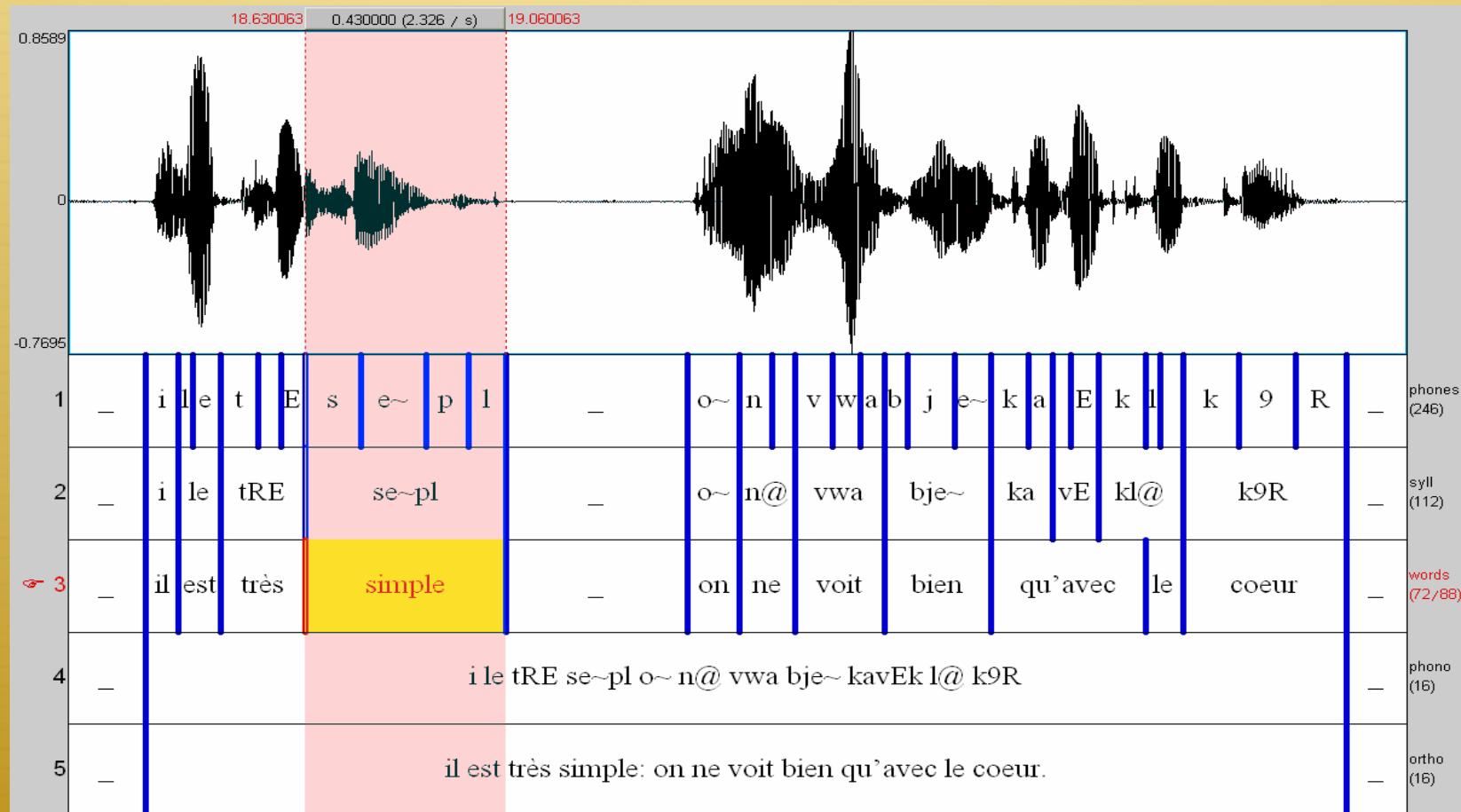
Alinieri.  
Sisteme complexe în IA

# În două dintre proiecte apare necesitatea de a produce alinieri



- ◆ P5. Recunoașterea optică a caracterelor pe tipărituri și manuscrise
  - ◆ intrarea constă din două fișiere: unul conținând o colecție de imagini de pagini scrise cu caractere chirilice; al doilea conținând transcrierile lor în caractere latine
  - ◆ programul vostru trebuie mai întâi să facă alinierea imaginilor cu textele, apoi să folosească aceste alinieri pentru a învăța să recunoască scrisul chirilic românesc
- ◆ P9. Aliniere voce-text:
  - ◆ intrarea constă din două canale: unul care conține un semnal vocal și al doilea care conține transcrierea textuală a înregistrării
  - ◆ programul vostru trebuie să producă un fișier XML care să marcheze alinierea celor două intrări

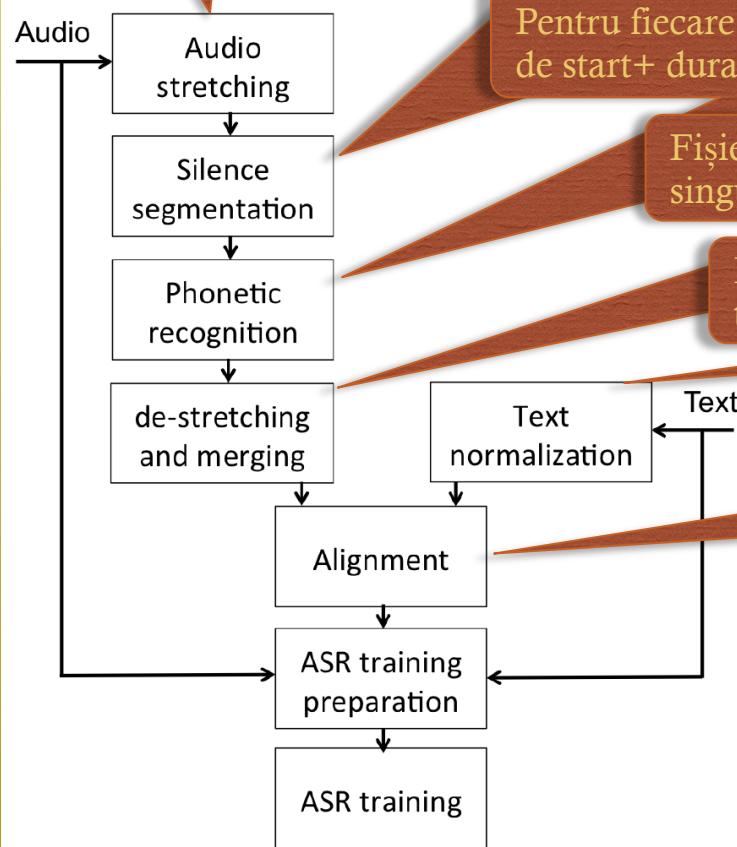
# Exemplu de aliniere



# Aliniere vorbire-text

Semnalul audio este întins cu un factor de 25% (a se vedea SoundTouch)

Pentru a depăsi probleme de memorie: în cazul în care sunetul întins este mai lung de 30 de secunde, se segmentează în fișiere audio individuale cu o lungime mai mică de 30 de secunde, după tăceri și, dacă există, vorbitori (v. LIUM).



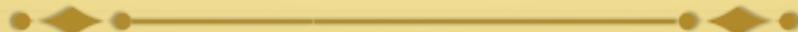
Pentru fiecare fonem se alege decodarea cu scorul maxim de încredere + momentul de start+ durata (v. Phnrec recognizer).

Fișierele individuale sunt întinse în timp și se unesc pentru a forma o singură transcriere fonetică.

Ieșirea: un set de simboluri grafice individuale + un simbol pentru tăcere.

Alinierea grafem-fonem (programare dinamică).

# Normalizarea textului



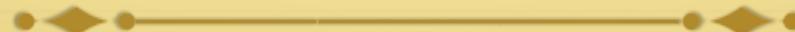
- ❖ Se folosesc grafeme pentru a nota sunetele limbii, în loc de alfabetul fonetic
  - ❖ se atinge aceeași calitate cu mai puține resurse
  - ❖ multe dintre grafeme reprezintă foneme (sunetele semnificative) ale limbii, cu excepții:
    - ❖ mai multe grafeme decât foneme:
      - ❖ în română: *che, chi, ghe, ghi*
      - ❖ în engleză: *sh*
    - ❖ mai puține grafeme decât foneme:
      - ❖ în rusă: *я*
  - ❖ un grafem pentru niciun sunet:
    - ❖ în engleză: *b* în *debt*
    - ❖ în spaniolă: *h* nu se pronunță

# Programare dinamică

- ❖ O secvență trebuie transformată în alta utilizând operațiuni de editare: înlocuire, inserare, ștergere.
  - ❖ Fiecare operație are un cost asociat, iar scopul este de a găsi o secvență a editărilor care să însumeze cel mai mic cost total.
  - ❖ Recursie: o secvență  $X$  este editată optim într-o secvență  $Y$  astfel:
    - ❖ ștergerea primului caracter al lui  $X$  și continuarea alinierii optime dintre coada lui  $X$  și  $Y$ ;
    - ❖ înlocuirea primul caracter al lui  $X$  cu primul caracter al lui  $Y$  și continuarea alinierii optime dintre coada lui  $X$  și coada lui  $Y$ ;
    - ❖ inserarea în  $X$  a primului caracter al lui  $Y$  și continuarea alinierii optime dintre  $X$  și coada lui  $Y$ .
- ❖ Alinierile parțiale pot fi memorate într-o matrice  $D$ , unde celula  $(i, j)$  conține costul alinierii optime a sirului  $X[1..i]$  la sirul  $Y[1..j]$ . Costul din celula  $(i, j)$  poate fi calculat prin adăugarea costului operațiunilor relevante în contextul curent la costul celulelor vecine și prin selectarea minimului.

$$D[X_i, Y_j] = \min \left\{ \begin{array}{l} pD(X_i) + D[X_{i-1}, Y_j], \text{ unde } pD(X_i) \text{ este penalizarea ștergerii grafemului } X_i \\ pC(X_i, Y_j) + D[X_{i-1}, Y_{j-1}], \text{ unde } pC(X_i, Y_j) = \text{penalizarea schimbării grafemului } X_i \text{ cu fonemul } Y_j \\ pI(Y_j) + D[X_i, Y_{j-1}], \text{ unde } pI(Y_j) \text{ este penalizarea inserării grafemului } Y_j \end{array} \right.$$

# Alinierea grafeme-foneme

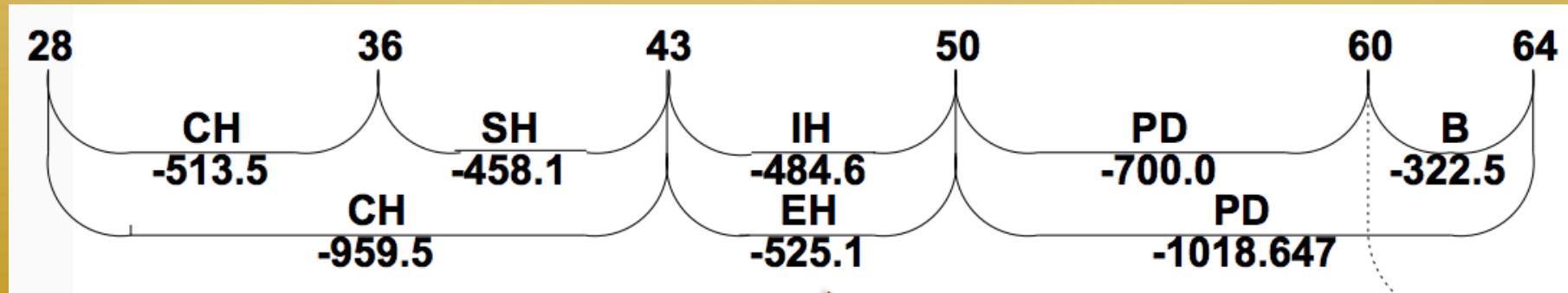


- ❖ Prin programare dinamică
  - ❖ se completează matricea distanțelor  $D$
  - ❖ se urmărește o cale în matrice, plecând din colțul dreapta-jos, înapoi spre colțul stânga-sus, pentru a afla secvența operațiilor de aliniere

	x	y	x	z	y
0	1	2	3	4	5
y	1	1	1	2	3
x	2	1	2	1	2
x	3	2	2	2	2
z	4	3	3	3	2

# Decoderul produce o latice

- Laticea fonemelor = un graf direcționat fără bucle
- Fiecare nod e etichetat cu un punct în timp, iar arcele cu foneme ipotetice și scorurile lor de recunoaștere

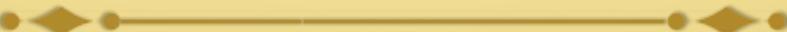


latice de grad 2 pentru  
cuvântul *ship*

# Terminologie de bază

- ❖ **fon (phone)** = un sunet de vorbire; fonurile sunt reprezentate prin simboluri fonetice (într-un limbaj de transcriere fonetică) derivate din litere ale unui alfabet
- ❖ **fonem (phoneme)** = (atât în lingvistică cât și în prelucrarea vorbirii) clasă abstractă pentru a surprinde similitudinea între diferențele pronunții ale unei litere
- ❖ **pitch** = perioada fundamentală a semnalului de vorbire. Reprezintă frecvența de vibrație a corzilor vocale în timpul producerii sunetelor (de exemplu, vocale)
- ❖ **diarizarea vorbitorului (speaker diarisation)** = procesul de împărțire a unui flux audio de intrare în segmente omogene în funcție de identitatea vorbitorului. O combinație între:
  - ❖ segmentarea vorbitorului: găsirea punctelor de schimbare a vocilor într-un flux audio
  - ❖ clusterizarea segmentelor individuale: gruparea segmentelor de vorbire pe baza caracteristicilor vocilor
- ❖ **model acustic** = relația dintre un semnal audio și fonemele sau alte unități lingvistice care alcătuiesc vorbirea; un model este învățat dintr-un set de înregistrări audio în pereche cu transcrierile lor; un program este antrenat pentru a crea reprezentări statistice ale sunetelor care alcătuiesc fiecare cuvânt
- ❖ **alfabet (writing script)** = set de caractere care codifică scrierile într-o limbă (latin, chirilic etc.)
- ❖ **dicționar de pronunție** = o colecție de reprezentări ortografice ale cuvintelor în pereche cu succesiunile corespunzătoare de foneme

# Alte probleme care necesită alinieri



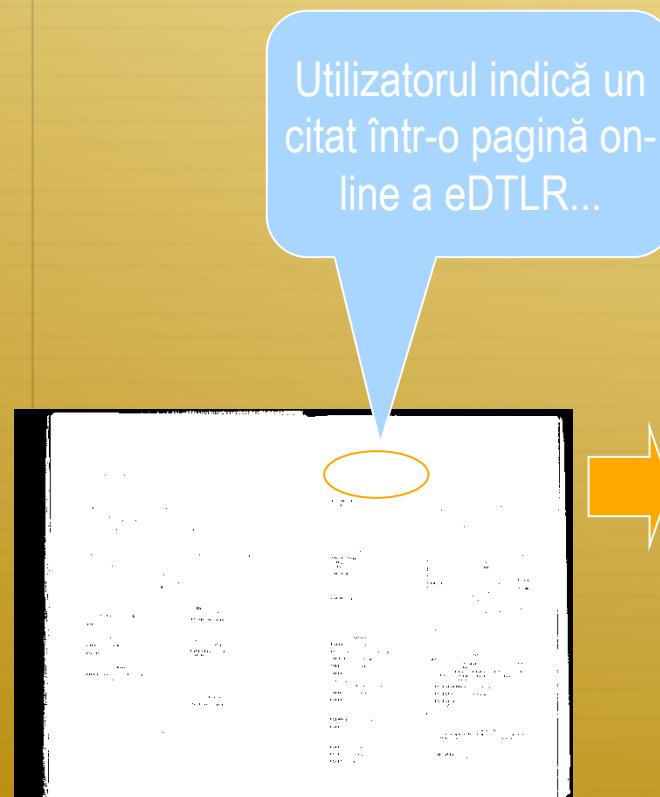
- ❖ Construirea eDTLR: formatul electronic al Dicționarului Tezaur al Limbii Române
  - ❖ citările din paginile corectate ale dicționarului sunt potrivite cu șirurile găsite prin OCR-izarea documentelor originale => vizualizează contextele de apariție ale citatelor din surse
- ❖ Identificarea de cuvinte cheie în vorbire, recuperarea mesajelor
  - ❖ cuvinte rostite sunt descompuse în trăsături; acestea sunt apoi utilizate pentru a potrivi întrebarea cu fluxul de vorbire
- ❖ Pagini OCR-izate din documente chirilice-românești sunt aliniate versiunilor corectate
  - ❖ din aceste alinieri pot fi deduse reguli de auto-corecție pentru OCR-izator

# Marele Dicționar tezaur al Limbii Române

## 1906 – 2010



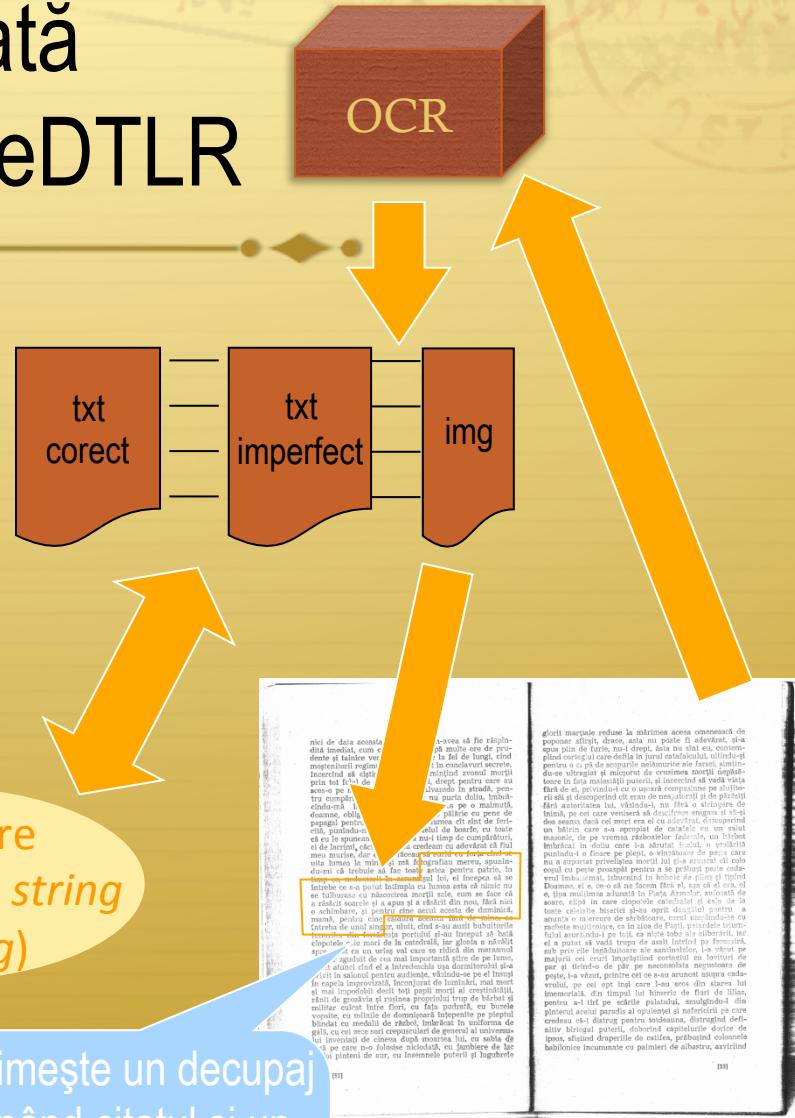
# Pentru ca utilizatorul să poată accesa sursa unui citat din eDTLR



Utilizatorul indică un citat într-o pagină online a eDTLR...

comparare  
(approximate string matching)

...și primește un decupaj conținând citatul și un context al lui în imaginea sursei originare.



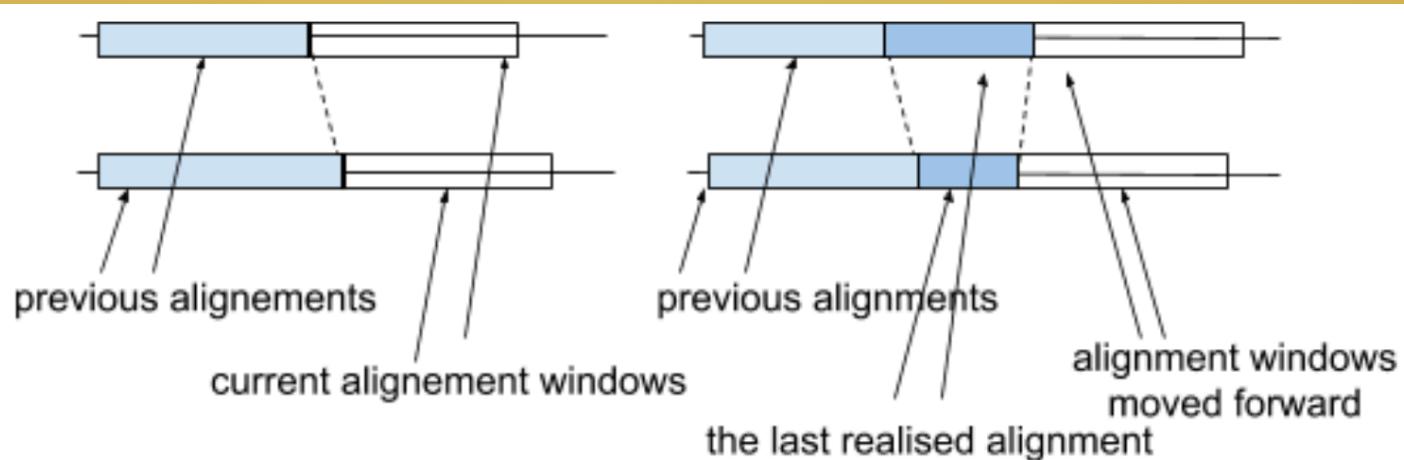
# Îmbunătățirea OCR-izatorului prin reguli de auto-corectare

- ❖ Cel mai adesea, software-ul OCR introduce erori. Îmbunătățirea rezultatului OCR se poate face în două moduri:
  - ❖ prin compararea rezultatelor cu o listă preexistentă de cuvinte
  - ❖ prin aplicarea unor **reguli de corecție deduse automat** => reguli deduse prin compararea ieșirii OCR (Test) cu aceeași întindere de text revizuită manual de experți (Gold)

# Alinierea fișierelor *Test* și *Gold*



- ❖ Alinierea poate fi obținută prin parcurgerea celor două texte de la stânga la dreapta, în paralel, cu ferestre de dimensiuni egale, căutând potriviri imperfecte în zone plasate între potriviri perfecte. Textul aliniat rămâne în spatele celor două ferestre.



# Generarea regulilor de corectare

- ❖ Diferențele găsite de mai multe ori semnalează erori repetitive ale OCR, care ar putea fi reduse prin reguli.
- ❖ Regulile de corectare contextuală trebuie să respecte formatul:  
$$<\text{Lctx}> <\text{Estr}> <\text{Rctx}> \Rightarrow <\text{Lctx}> <\text{Cstr}> <\text{Rctx}>$$
unde: Lctx = contextul stâng, Rctx = contextul drept,  
Estr = sir eronat, Cstr = sir corect.
- ❖ Pentru ca o regulă să fie validă, în perechea de texte Test-Gold, trebuie să fie îndeplinite condițiile:
  - ❖ oricând sirul  $<\text{Lctx}> <\text{Estr}> <\text{Rctx}>$  este în Test, Gold conține în aceeași poziție sirul  $<\text{Lctx}> <\text{Cstr}> <\text{Rctx}>$
- ❖ Regulile de corecție ar trebui să includă contextul minimal care nu produce corecții false => mărirea progresivă a  $<\text{Lctx}>$  și  $<\text{Rctx}>$  până cee această condiție este îndeplinită (cu pericolul ca unele situații de corecție să fie pierdute, deoarece un context mai mare face ca acesta să fie mai restrictiv).

# leșirea aliniatorului



- ❖ O reprezentare posibilă:
  - ❖ D[c] marchează **Ștergerea** unui caracter c;
  - ❖ I[c] marchează **Inserarea** unui caracter c;
  - ❖ C[c,d] marchează **Schimbarea** unui caracter c într-un caracter d;
  - ❖ orice alt caracter (inclusiv spațiile și semnele de punctuație) reprezintă caractere neschimilate

Test

Gold

osăndă au făcut ^să fie aduși (^și^ făcătorii  
os\$ndă au făcut să fie aduși și făcătorii de reale.

Cesă^zicăcei ce pre Dmnul carele iau făcut pre dănsii ,  
Ce să zică cei ce pre Dmnul carele iau făcut pre d\$nsii ,

Test

Gold

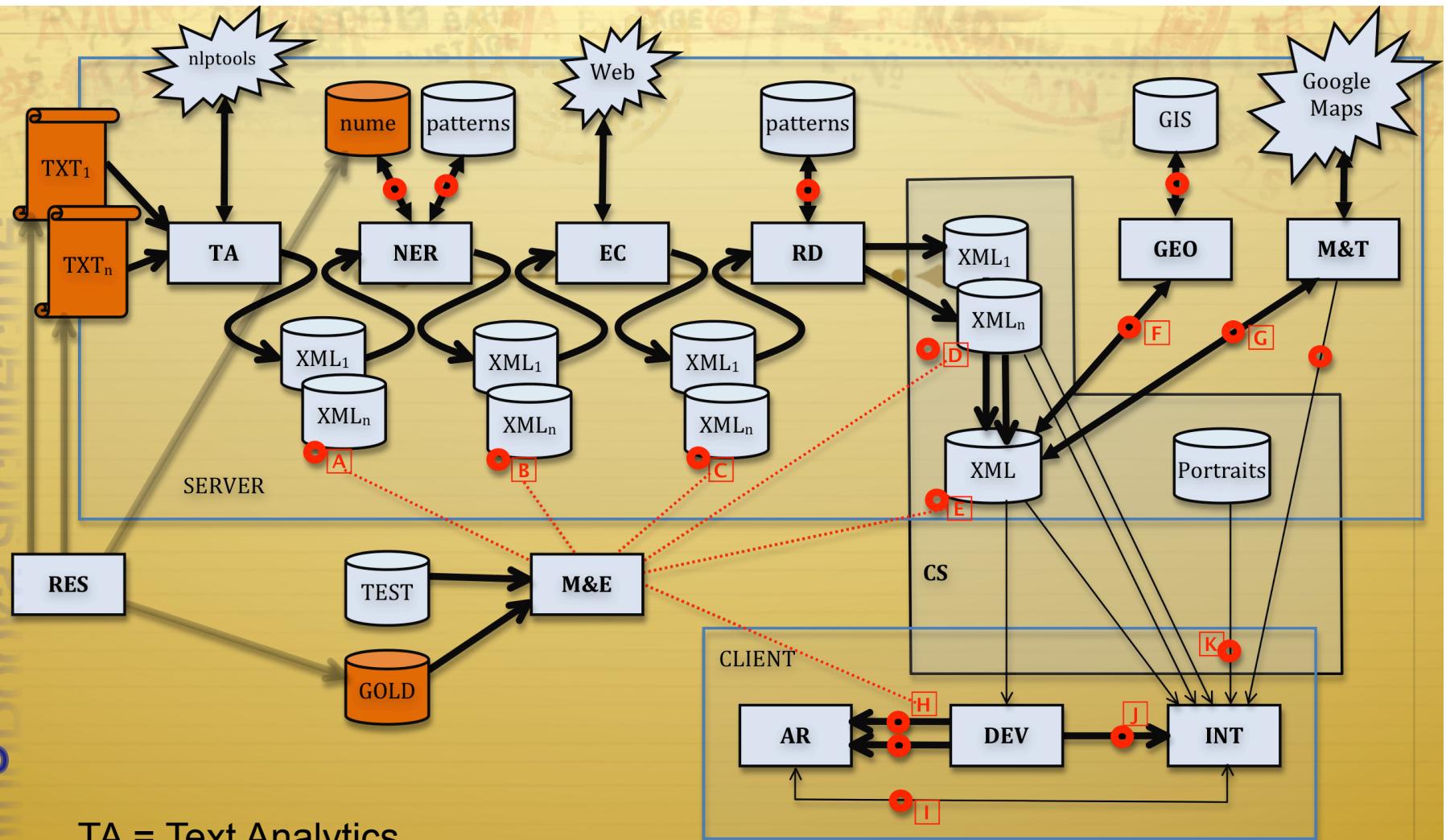
o-s-C[ă,\$]-n-d-ă- -a-u- -f-ă-c-u-t -D[^]-s-ă- -f-i-e- -a-  
d-u-ş-i-D[()]-D[^]-ş-i-D[^]- -f-ă-c-ă-t-o-r-i-i-I[d]-I[e]-  
I[ ]-I[r]-I[e]-I[a]-I[l]-I[e]-I[.]- -C-e-I[ ]-s-ă-C[^, ]-  
z-i-c-ă-I[ ]-c-e-i- -c-e- -p-r-e- -D-m-n-u-l- -c-a-r-e-l-  
e- -i-a-u- -f-ă-c-u-t- -p-r-e- -d-C[ă,\$]-n-ş-i-i- -, -



• ◆ • — — — ◆ •

# Cum să construim sisteme complexe?

# MappingBooks architecture



TA = Text Analytics

NER = Name Entity Recognition

EC = Entity Crowling

RD = Relations Detection

GEO = Geography

M&T = Maps and Trajectories

AR = Augmented Reality

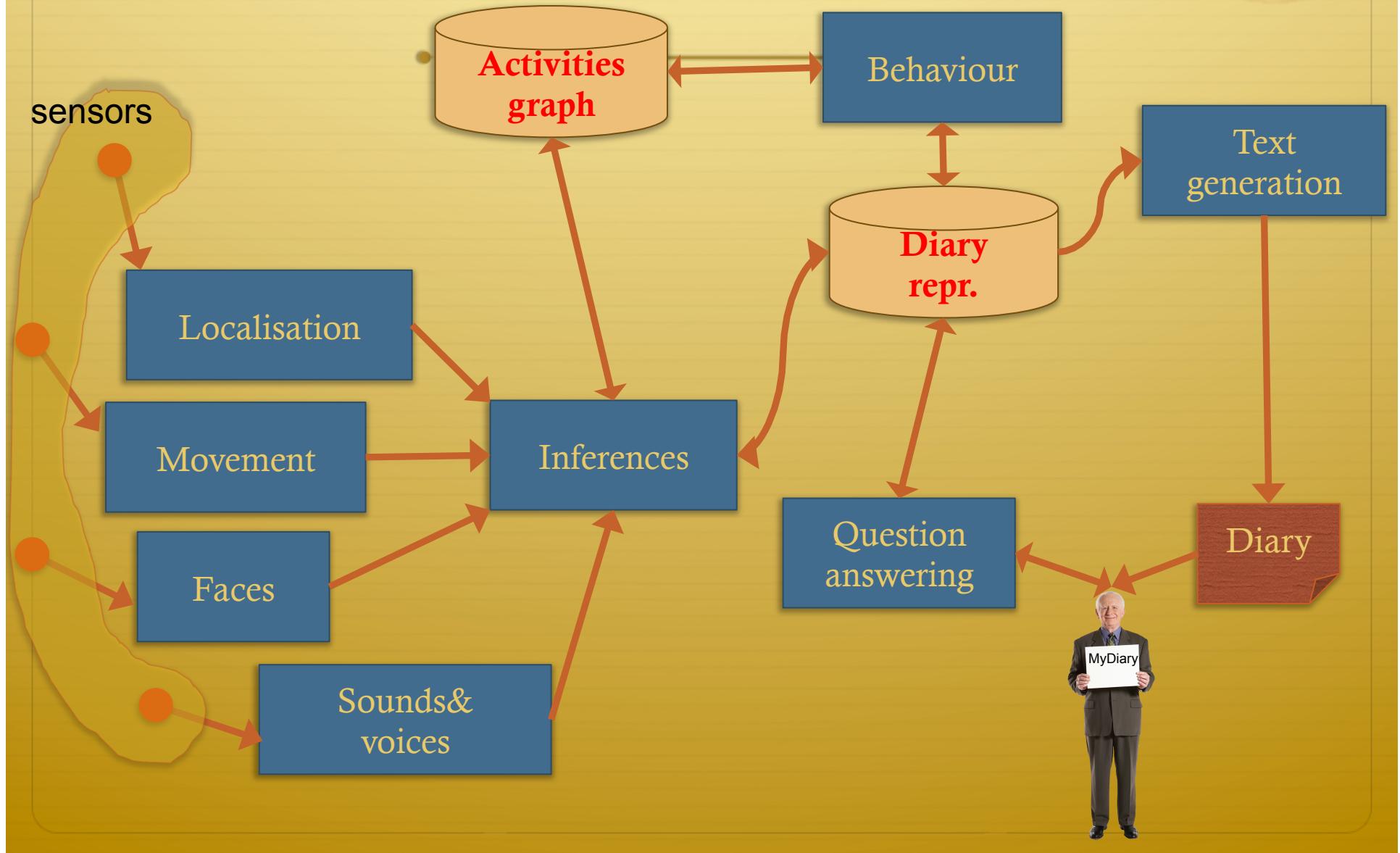
DEV = Device Info

INT = Interfaces

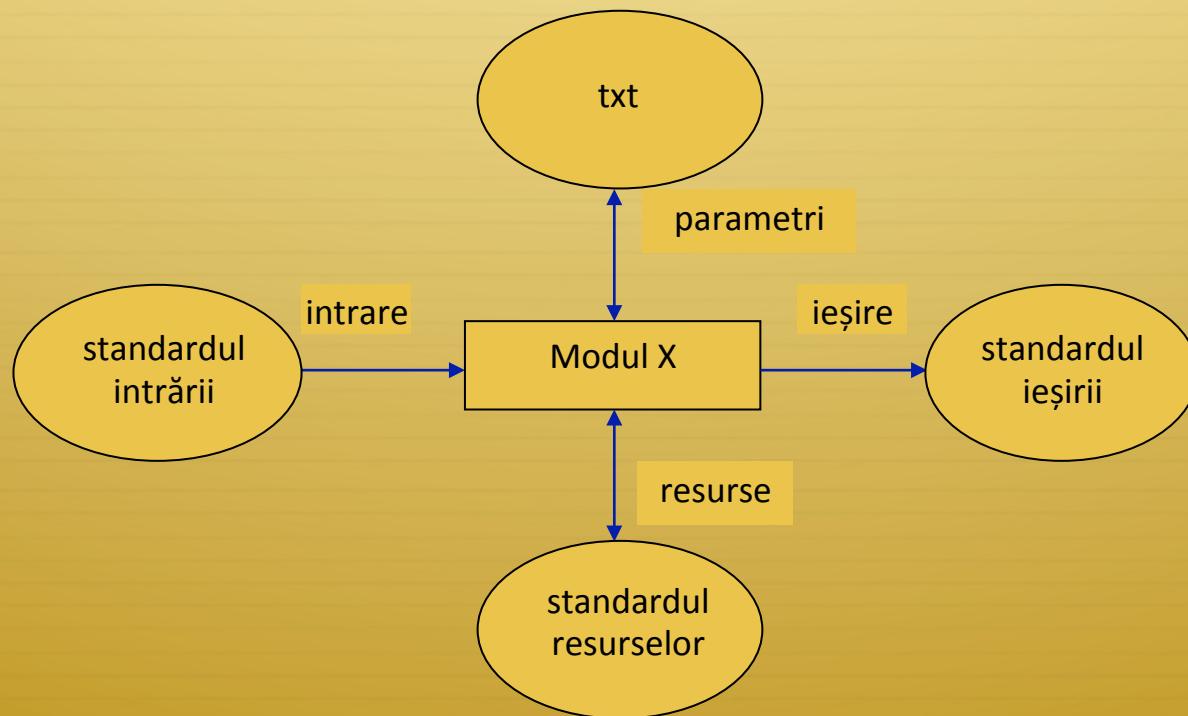
RES = Resources

M&E = Management and Evaluation

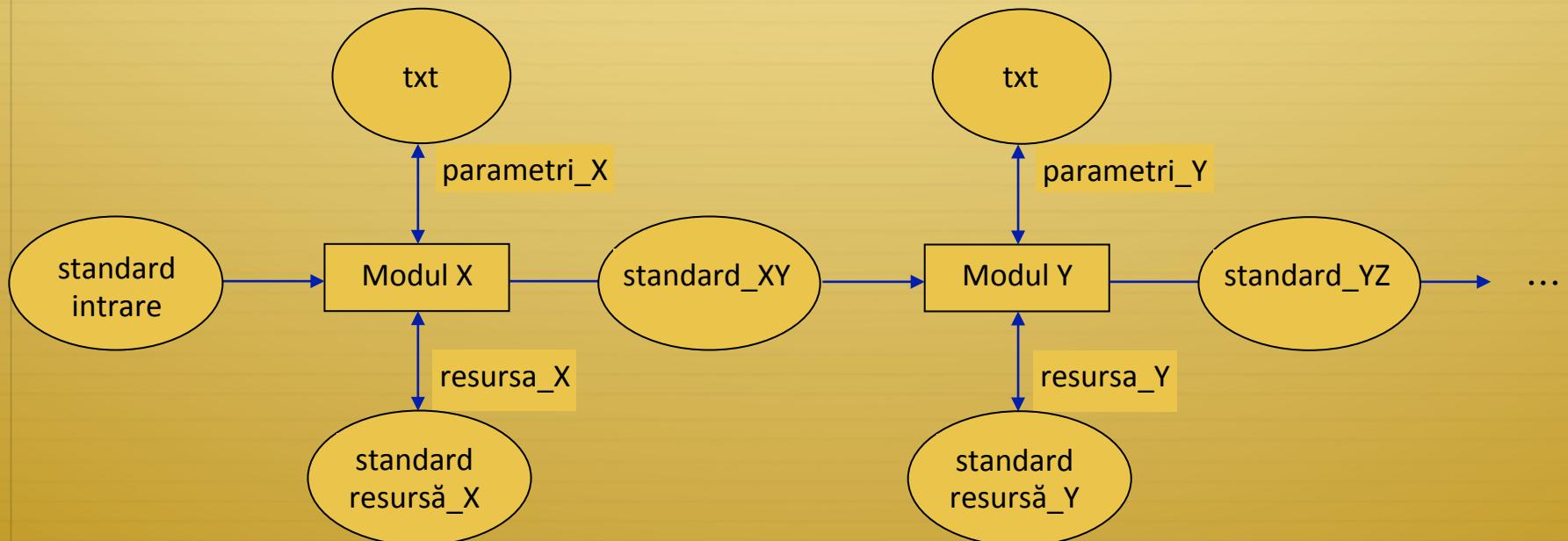
# MyDailyLife – schema generală



# Forma recomandată a unui modul care participă într-un lanț de prelucrări



# Dacă se înțeleg în intrări/iesiri, modulele pot fi înlántuite



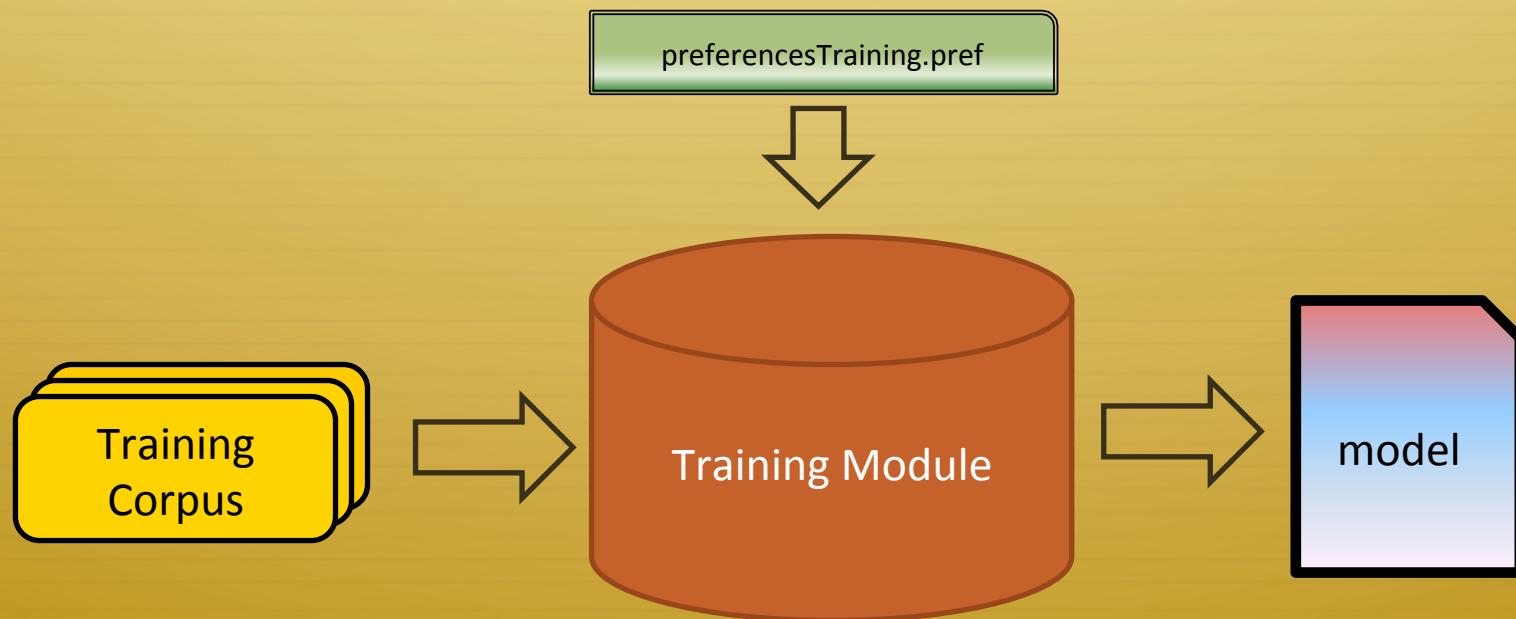
# Cum se calibrează un modul?



- ❖ Să presupunem că vrem să construim un modul care să realizeze un anumit obiectiv. Atunci, de fapt, va trebui să fabricăm 3 module:
  - **Modulul de antrenare (TM)**
  - **Modulul propriu-zis (X)**
  - **Modulul de evaluare (EM)**

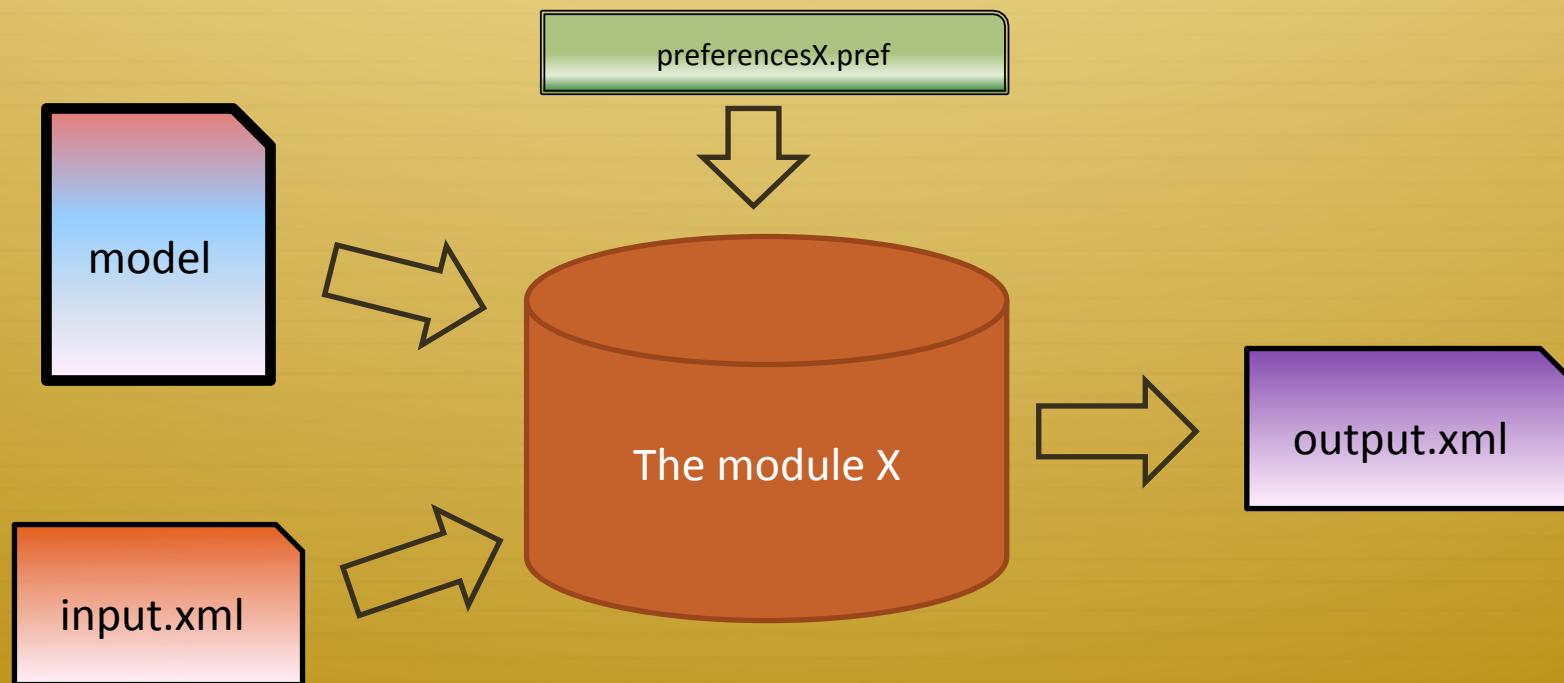
# Modulul de antrenare (TM)

- ★ TM extrage dintr-un corpus de antrenare un **model** care va fi apoi folosit de modulul X.



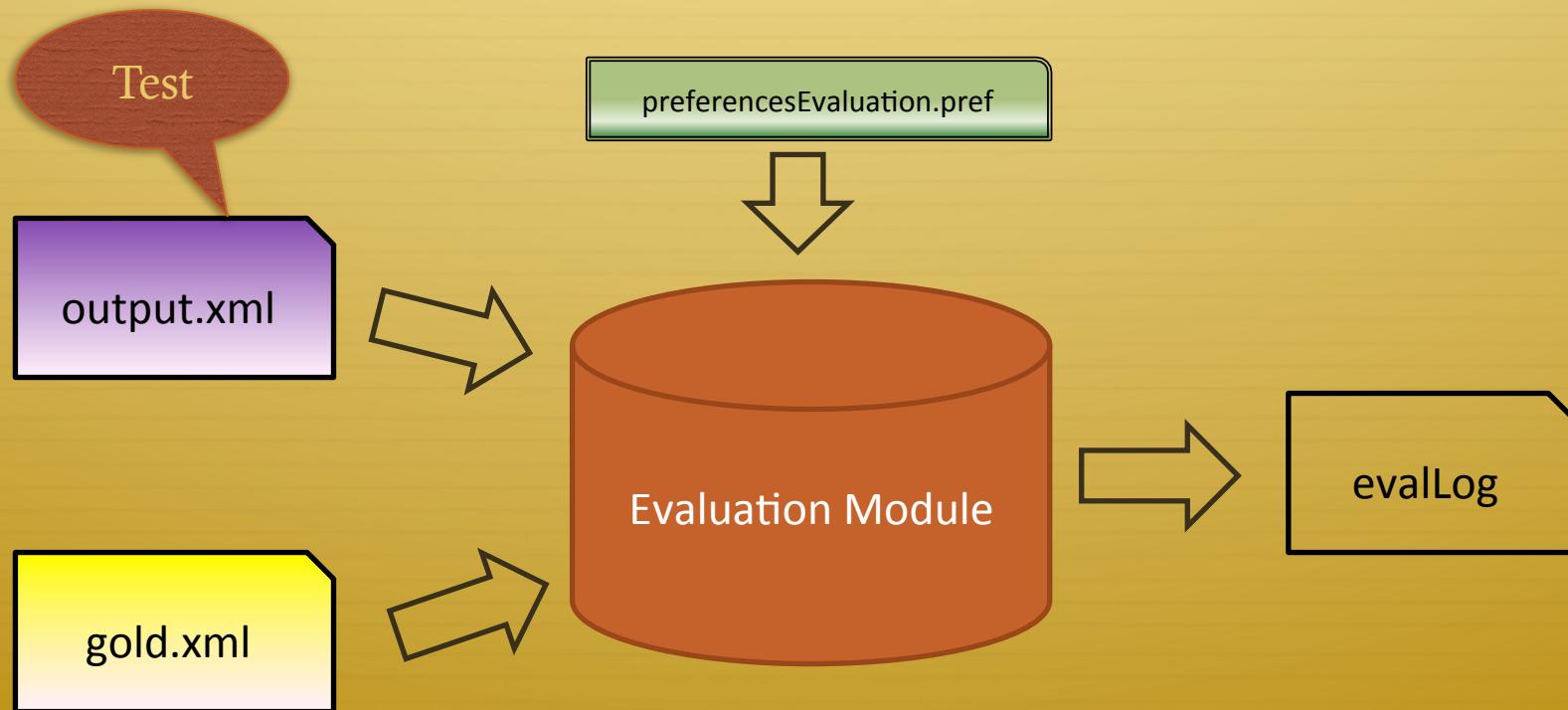
# Modulul X

- ★ X aplică un algoritm asupra unei *intrări* pentru a o transforma în conformitate cu *modelul* învățat.

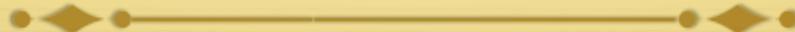


# Modulul de evaluare (EM)

- ★ EM evaluează (compară) un fișier *Test* față de un fișier considerat corect (de aur) *Gold*.

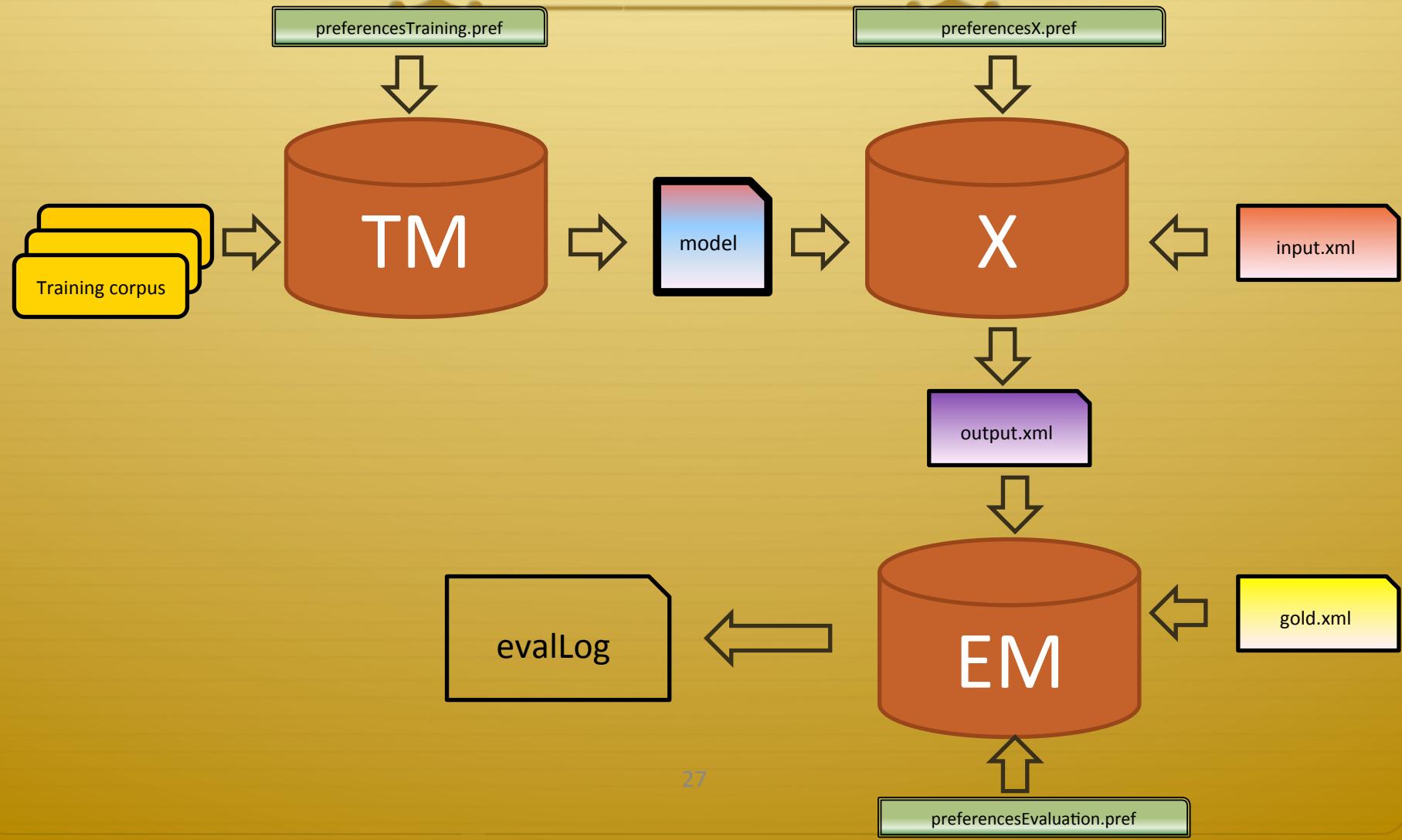


# Măsuri în evaluare

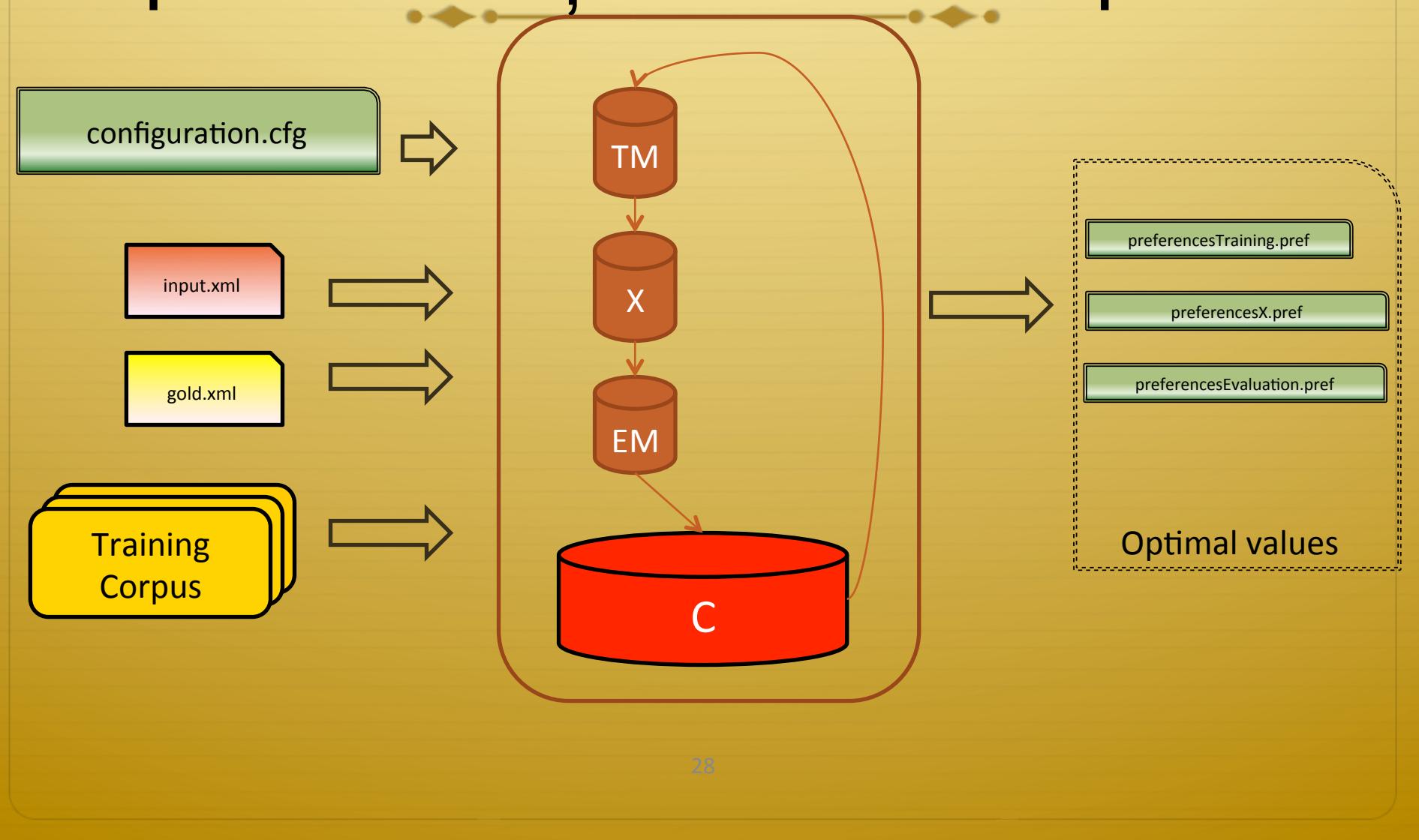


- Precision =  
 $\# \text{itemi în comun în Test & Gold} / \# \text{itemi în Test}$
- Recall =  
 $\# \text{itemi în comun în Test & Gold} / \# \text{itemi în Gold}$
- F-measure =  $2 * P * R / (P + R)$

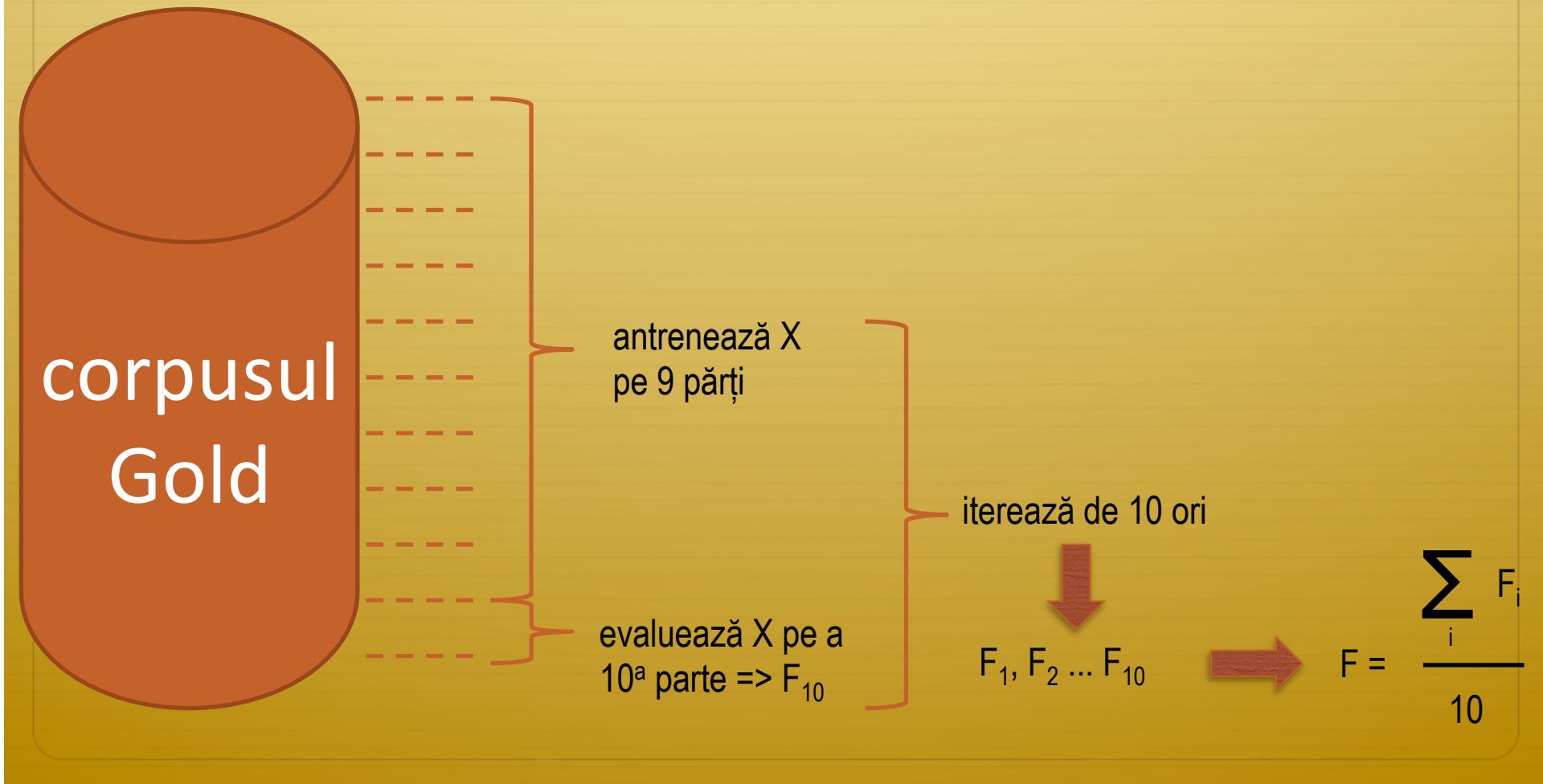
# Asamblarea unui sistem de calibrare



# Calibrare = iterarea parametrilor până la obținerea unui optim



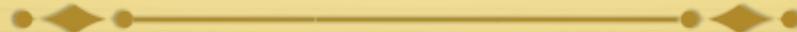
# Evaluarea unu-din zece (10-fold)



# Bibliografie

- ❖ An implementation of the SOLA (Synchronous-OverLap-Add) algorithm can be found in SoundTouch Audio Processing Library:  
<http://www.surina.net/soundtouch>
- ❖ Xavier Anguera, Jordi Luque, Ciro Gracia (2014). Audio-to-text alignment for speech recognition with very limited resources, in proc. Interspeech 2014, Singapore. Available:  
[http://www.xavieranguera.com/papers/IS2014\\_phonealignment.pdf](http://www.xavieranguera.com/papers/IS2014_phonealignment.pdf)
- ❖ M. Rouvier, G. Dupuy, P. Gay, E. Khoury, T. Merlin, and S. Meignier (2013). “An Open-source State-of-the-art Toolbox for Broadcast News Diarization,” in Proc. Interspeech. Available: <http://www-lium.univ-lemans.fr/diarization>.
- ❖ P. Schwarz, “Phoneme Recognition Based on Long Temporal Context,” Ph.D. dissertation, 2008. Available:  
<http://speech.fit.vutbr.cz/software/phoneme-recognizerbased-long-temporal-context>.
- ❖ M. Killer, S. Stüker, and T. Schultz, “Grapheme Based Speech Recognition,” in Eurospeech, pp. 3141–3144.

# Site-uri folositoare



- ❖ The open source SoundTouch audio processing library



## SoundTouch Audio Processing Library

The latest stable release is 2.1.1 released on 15-November-2018. See [README](#) for release details and what's new in this release.

### About the SoundTouch library

#### SoundTouch

[About SoundTouch](#)

[SoundStretch utility](#)

[Download](#)

[Source codes](#)

[Audio Examples](#)

[Readme](#)

[F.A.Q.](#)

[License](#)

Do you like SoundTouch? Then please support SoundTouch development and maintaining by a donation!

[Make A Donation](#)

Unrestricted Royalty-free license also

The SoundTouch Library Copyright © Olli Parviainen 2001-2018

SoundTouch is an open-source audio processing library for changing the Tempo, Pitch and Playback Rates of audio streams or audio additionally supports estimating stable beats-per-minute rates for audio tracks.

- **Tempo (time stretch)**: Changes the sound to play at faster or slower tempo than originally without affecting the sound pitch.
- **Pitch (key)** : Changes the sound pitch or key while keeping the original tempo (speed).
- **Playback Rate** : Changes both tempo and pitch together as if a vinyl disc was played at different RPM rate.

The SoundTouch library is intended for application developers writing sound processing tools that require tempo/pitch control functions playing around with the sound effects.

The SoundTouch library source kit includes also an example utility [SoundStretch](#) for processing .wav audio files from command-line

**Access** the [source codes in gitlab](#) or **download** the compiled [executables](#).

### More information:

- [Example sound clips](#) of each tempo/pitch/rate control mode
- [List of applications that use SoundTouch library](#)
- [Frequently Asked Questions](#)
- Theory behind how SoundTouch works - [read tutorial on audio time/pitch scaling basics](#)

# Site-uri folositoare

- ❖ The LIUM Speaker Diarization toolkit

## LIUM Speaker Diarization Wiki

Welcome

Overview

Download

Contact

Related projects

Usage

Quick start

Licence

Data

Diarization

Acoustic feature

Models

Tools

Commun parameter

Programmes

Scripting

Feature set

Gaussian / GMM training

Speaker Identification

Gender detection

Single-show Diarization

Cross-show Diarization

I-Vector

Programming

Data structure

Algorithm

HowTo

whish List

## Welcome

This wiki presents the LIUM\_SpkDiarization tools. LIUM\_SpkDiarization is a software dedicated to speaker diarization (ie speaker segmentation and clustering). It is written in Java, and includes the most recent developments in the domain.

LIUM\_SpkDiarization comprises a full set of tools to create a complete system for speaker diarization, going from the audio signal to speaker clustering based on the CLR/NCLR metrics. These tools include MFCC computation, speech/non-speech detection, and speaker diarization methods.

This toolkit was developed for the French ESTER2 evaluation campaign, where it obtained the best results for the task of speaker diarization of broadcast news in 2008<sup>[1]</sup>. Please note that the toolbox is optimized for radio or tv shows. You should not expect the same level of performances on phone conversations and meetings.

### Some related publications

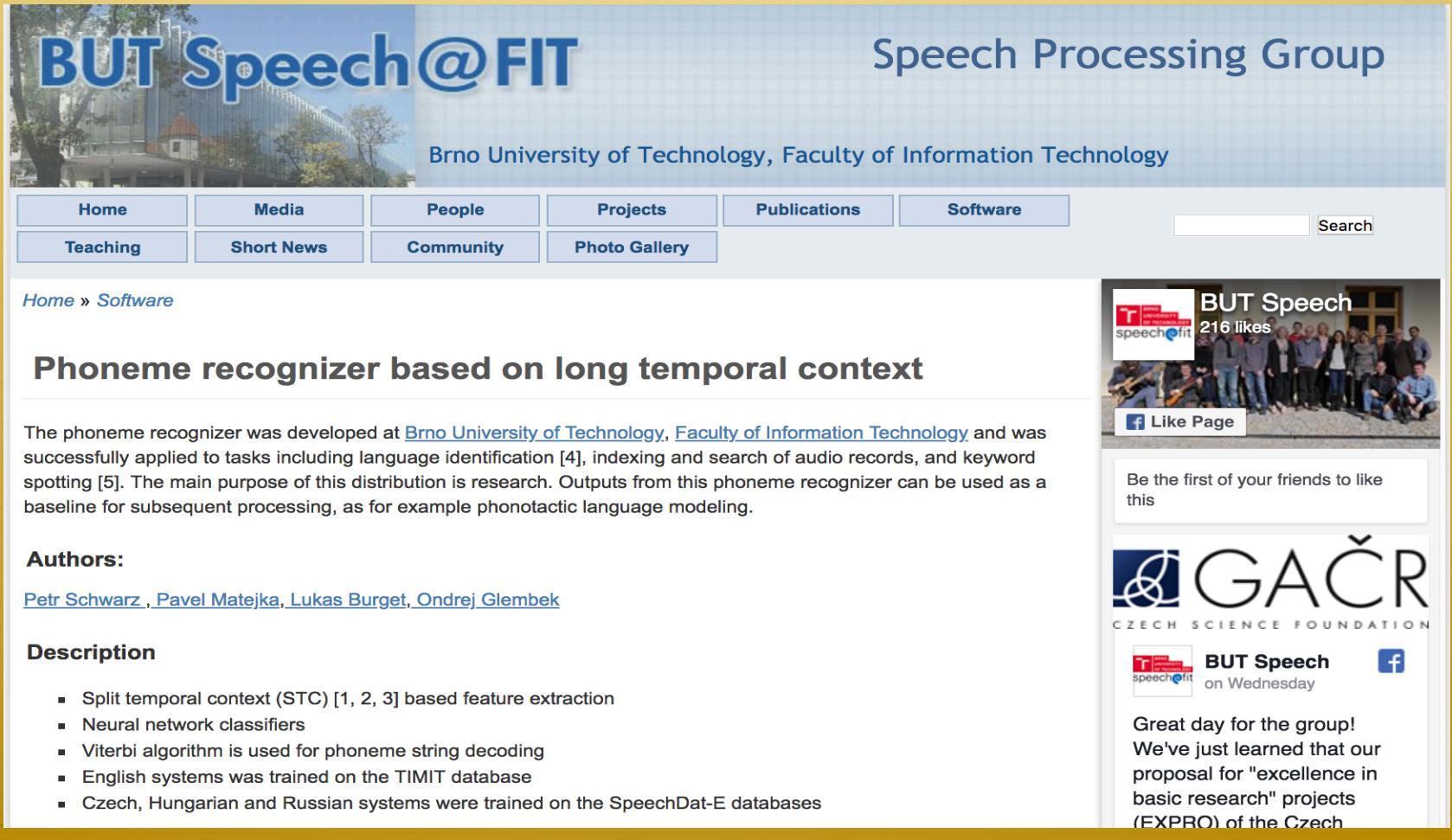
If you are using this toolkit in your research please cite one of these papers.

#### Speaker Diarization

- M. Rouvier, G. Dupuy, P. Gay, E. Khoury, T. Merlin, S. Meignier, "An Open-source State-of-the-art Toolbox for Broadcast News Diarization," Interspeech, Lyon (France), 25-29 Aug. 2013
- [toolkit-interspeech2013.pdf](#)
- S. Meignier, T. Merlin, "LIUM SpkDiarization: An Open Source Toolkit For Diarization," in Proc. CMU SPUD Workshop, March 2010, Dallas (Texas, USA).
- [diarization-cmu-spud-2010.pdf](#)

# Site-uri folositoare

## ❖ Phoneme recognition



The screenshot shows the homepage of the BUT Speech@FIT website. The header features the logo "BUT Speech@FIT" and the text "Speech Processing Group" and "Brno University of Technology, Faculty of Information Technology". Below the header is a navigation menu with links: Home, Media, People, Projects, Publications, Software, Teaching, Short News, Community, and Photo Gallery. A search bar is also present. The main content area displays a news article titled "Phoneme recognizer based on long temporal context". The article discusses the development of a phoneme recognizer at Brno University of Technology, Faculty of Information Technology, which was successfully applied to tasks like language identification, indexing audio records, and keyword spotting. It highlights the recognizer's use as a baseline for subsequent processing. Below the article, there are sections for "Authors" (Petr Schwarz, Pavel Matejka, Lukas Burget, Ondrej Glemek) and "Description", which lists features such as split temporal context (STC), neural network classifiers, Viterbi algorithm for decoding, and training on TIMIT and SpeechDat-E databases. To the right of the main content, there is a sidebar with a Facebook-like interface showing a group photo, a 'Like Page' button, and a message encouraging users to like the page. At the bottom right, there is a logo for GAČR (Czech Science Foundation) and a message about a successful proposal submission.

**BUT Speech@FIT**  
Speech Processing Group  
Brno University of Technology, Faculty of Information Technology

Home Media People Projects Publications Software  
Teaching Short News Community Photo Gallery

Search

Home » Software

## Phoneme recognizer based on long temporal context

The phoneme recognizer was developed at [Brno University of Technology, Faculty of Information Technology](#) and was successfully applied to tasks including language identification [4], indexing and search of audio records, and keyword spotting [5]. The main purpose of this distribution is research. Outputs from this phoneme recognizer can be used as a baseline for subsequent processing, as for example phonotactic language modeling.

**Authors:**

[Petr Schwarz](#), [Pavel Matejka](#), [Lukas Burget](#), [Ondrej Glemek](#)

**Description**

- Split temporal context (STC) [1, 2, 3] based feature extraction
- Neural network classifiers
- Viterbi algorithm is used for phoneme string decoding
- English systems was trained on the TIMIT database
- Czech, Hungarian and Russian systems were trained on the SpeechDat-E databases

**BUT Speech**  
216 likes

Be the first of your friends to like this

**GAČR**  
CZECH SCIENCE FOUNDATION

**BUT Speech**  
on Wednesday

Great day for the group!  
We've just learned that our  
proposal for "excellence in  
basic research" projects  
(EXPRO) of the Czech