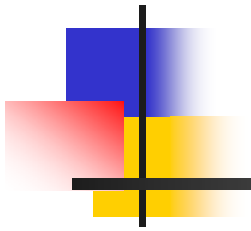
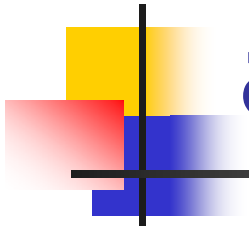


Building stemmers for Information Retrieval and related domains

Nikitas N. Karanikolas,
Department of Informatics,
Technological Educational Institute (TEI) of Athens, Greece,
nnk@teiath.gr

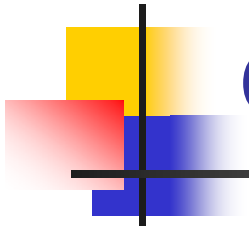
University of Novi Sad (НОВИ САД) – May 2017



about stemmers

- They are used in various text processing tasks: search engines, document/text summarizers, document/text classifiers, etc,
- Stemmers produce normalized forms of words in order to handle as one attribute all the inflected word-forms existing in documents for the same word,
- Alternative solution is the usage of lemmatizers that conflate a set of words in their etymological root.

Stemmer's and Lemmatizer's examples



Greek

- Τράπεζα (Bank),
- Τράπεζες (Banks),
- Τραπεζικές (Banking),
- Τραπεζική (Banking)

- Stemmer's result: ΤΡΑΠ,
- Lemmatizer's result: ΤΡΑΠΕΖΑ

Albanian

- PROVË
- PROVOHEJ
- PROVONTE
- PROVUAR

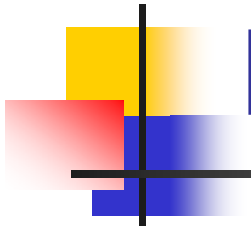
- Stemmer's result: PROV,
- Lemmatizer's result: PROVË

Stemming example for Serbian

	singular	plural
nominative	вòда	вòде
genitive	вòде	вóдā
dative	води	водама
accusative	воду	воде
vocative	водо	воде
locative	води	водама
instrumental	водом	водама

To complicate more: водица

The stem could be **вод**

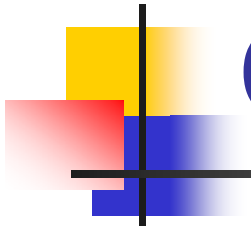


Rule based stemmers

- Porter's stemmer uses five levels (granularity, different rules in each level)
- Lovin's stemmer uses 2 steps (suffix elimination and recording step)
- Paice's stemmer is an iterating algorithm using the same rules in each step

The purpose of research

- The domain of interest is the creation of a stemmer, when the development team does not have knowledge of the target language of stemmer.
- Our approach requires two resources:
  - a list of available suffixes used in the target language and
  - a training set of words in the target language with their translations in the native language of the experts.
- Both resources can be easily constructed by speakers of both languages (target and experts' native language).
- Speakers of both languages are needed to have a secondary or high school level (no university degree).

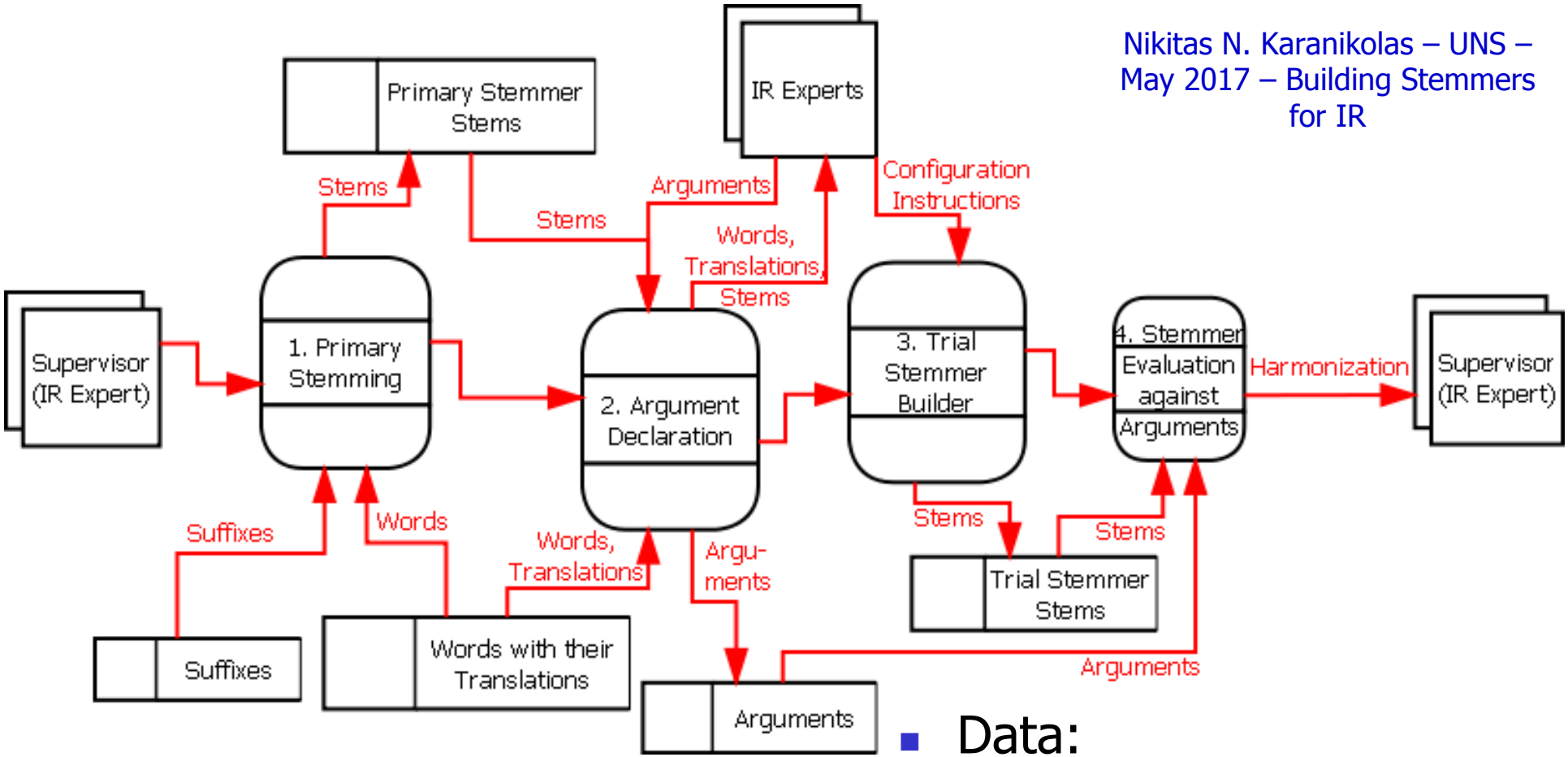


Overview of Approach

- The approach assumes a very simple (primary or bootstrapping) stemmer that provides stems by simply removing the longer suffix that match with a given word.
- Experts express their arguments regarding the results of the primary stemmer.
- The final step is a trial and error approach that permits to an IR (information retrieval) expert to dynamically construct a better stemmer, without coding even a single line of code.

Approach in Data Flow Diagram

Nikitas N. Karanikolas – UNS –
May 2017 – Building Stemmers
for IR



- Processes:
  - 1. Primary Stemmer
  - 2. Argument Declaration
  - 3. Stemmer Builder
  - 4. Evaluator
- Data:
  - Suffixes
  - Words with Translations
  - Arguments
  - Primary Stemmer's Stems
  - Trial Stemmer Stems

Examples of Experts' argumentation (1/3)

Ref. num	Word	Stem	Translation	Argument
1489	HIMARË	HIM	οι ύμνοι	CS (HIM)
1490	HIMNET	HIMN	των υμνών	
1491	HIMNI	HIM	ο ύμνος	
1492	HIMNIN	HIM	τον ύμνο	
1493	HIMNIT	HIM	του ύμνου	

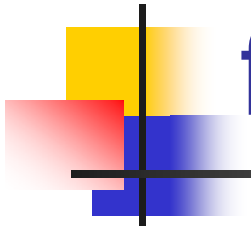
Examples of Experts' argumentation (2/3)

Ref. num	Word	Stem	Translation	Argument
1963	KOSTA	KOST	όνομα ανθρώπου	DS
1964	KOSTON	KOST	κοστίζει	

Examples of Experts' argumentation (3/3)

Ref. num	Word	Stem	Translation	Argument	
3172	PËRBEHEJ	PËRBE	αποτελείται	DS	CS_1
3173	PËRBËJNË	PËRBË	αποτελούνται		
3174	PËRBËN	PËRB	αποτελεί		
3175	PËRBËNTE	PËRBË	αποτελούσε		
3176	PËRBËRË	PËRBËR	αποτελείται		
3177	PËRBËRJE	PËRBËR	σύνθεση		CS_2
3178	PËRBËRJEN	PËRBËR	η σύνθεση		

Kinds of arguments and facilities



- Kinds
  - Complaints
  - Verifications
  - Why expressing verifications
- Facilities
  - Movements
  - Rules for x in CS(x)

Complaints - CS

Ref.Num.	Word	Stem	Translation	Argument
3562	PRONARET	PRONAR	Ιδιοκτήτες	CS (PRON)
3563	PRONAREVE	PRONAR	των ιδιοκτητών	
3564	PRONAVE	PRON	των ιδιοκτησιών	
3565	PRONË	PRO	ιδιοκτησία	
3566	PRONES	PRON	της ιδιοκτησίας	

Complaints – DS/CS

Ref.Num.	Word	Stem	Translation	Argument	
2049	KUKËS	KUK	πόλη της Αλβανίας	DS	CS_1
2050	KUKËSIT	KUK	της πόλης αυτής		
2051	KUKULL	KUK	κούκλα		CS_2

Complaints – DS/CS

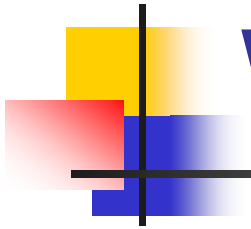
Ref.Num.	Word	Stem	Translation	Argument	
1059	FILL	FILL			CS_1
1060	FILLIM	FILL		DS	CS_2
1061	FILLIMI	FILL			
1062	FILLIMIN	FILL			
1063	FILLIMISHT	FILL			
1064	FILLIMIT	FILL			
1065	FILLOI	FILL			
1066	FILLOJ	FILL			
1067	FILLOJMË	FILL			
1068	FILLOVA	FILL			
1069	FILLUA	FILL			
1070	FILLUAN	FILL			
1071	FILLUAR	FILL			

Verifications - CS

Ref.Num.	Word	Stem	Translation	Argument
1073	FILOZOFËT	FILOZOF	οι φιλόσοφοι	CS (FILOZOF)
1074	FILOZOFINË	FILOZOF	τον φιλοσοφισμό	
1075	FILOZOFISË	FILOZOF	της φιλοσοφίας	

Verifications – DS/CS

Ref.Num.	Word	Stem	Translation	Argument	
176	ARMATA	ARMAT	στρατός	DS	CS_1
177	ARMATËS	ARMAT	του στρατού		
178	ARMATOSUR	ARMATOS	οπλισμένος		CS_2
179	ARMATOSURA	ARMATOS	οπλισμένα		
180	ARMË	ARM	όπλα		CS_3
181	ARMËT	ARM	τα όπλα		
182	ARMËVE	ARM	των όπλων		



Why expressing verifications

- The need to emphasize or verify the results of the primary stemmer comes from the algorithm used to compare the harmonization of a given stemmer with the expert's arguments.
- The matching factor (in an off hand simplification) is calculated as the number of experts arguments (CS and DS/CS) that are verified by the stemmer's results (stems), normalized by the number of arguments.
- The rest stemmer's results (stems that correspond to words which are outside the experts' arguments) contribute only slightly to the matching factor.
- The criterion for a stem outside the experts' arguments to contribute (increase slightly the matching factor) is that it differs from its adjacent ones.
- This requirement/criterion is the only difference against some earlier version.

Reordering & Complaints – CS

Ref.Num.	Word	Stem	Translation	Argument
1554	IDENTIFIKUARA	IDENTIFIK	προσδιορισμένα	∅
1552	IDEJA	IDE	η ιδέα	CS(IDE)
1553	IDENË	ID	την ιδέα	
1555	IDEVE	ID	των ιδεών	

Reordering & Complaints– DS/CS

Ref.Num.	Word	Stem	Translation	Argument	
3511	PRILL	PRI	Απρίλιος	DS	CS_1
3523	PRISJA	PRIS	περίμενα	DS	CS_2
3524	PRISNIN	PRIS	περίμεναν	DS	CS_2
3525	PRITËN	PRIT	περίμεναν	DS	CS_2
3526	PRITET	PRI	αναμένεται	DS	CS_2
3527	PRITJEN	PRIT	την αναμονή	DS	CS_2
3528	PRITUR	PRIT	φιλόξενος	DS	CS_2

Reordering & Verifications – CS

Ref.Num.	Word	Stem	Translation	Argument
3576	PROVINCË	PROVINC	επαρχία κράτους	∅
3575	PROVË	PROV	δοκιμή	CS(PROV)
3577	PROVOHEJ	PROV	δοκιμάζονταν	
3578	PROVONTE	PROV	δοκίμαζε	
3579	PROVUAR	PROV	δοκιμασμένο	

x in CS(x) – example

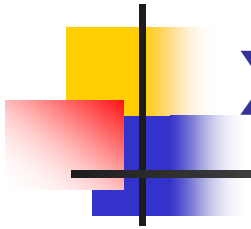
Ref.Num.	Word	Stem	Translation	Argument
2014	KRYER	KR	εκτελεσμένος	CS (KRYER)
2015	KRYERA	KRYE	το εκτελεσμένο	
2016	KRYERJEN	KRYER	την εκτέλεση	

KRYER:
exist in every
longest
most frequent

x in CS(x) – example

Ref.Num.	Word	Stem	Translation	Argument
2063	KUNDER	KUND	κατά	CS (KUND)
2064	KUNDËRSHTIM	KUNDËRSHT	ένσταση	
2065	KUNDËRSHTIVE	KUNDËRSHT	εναντιώθηκες	
2066	KUNDËRSHTUAN	KUNDËRSHT	εναντιώθηκαν	

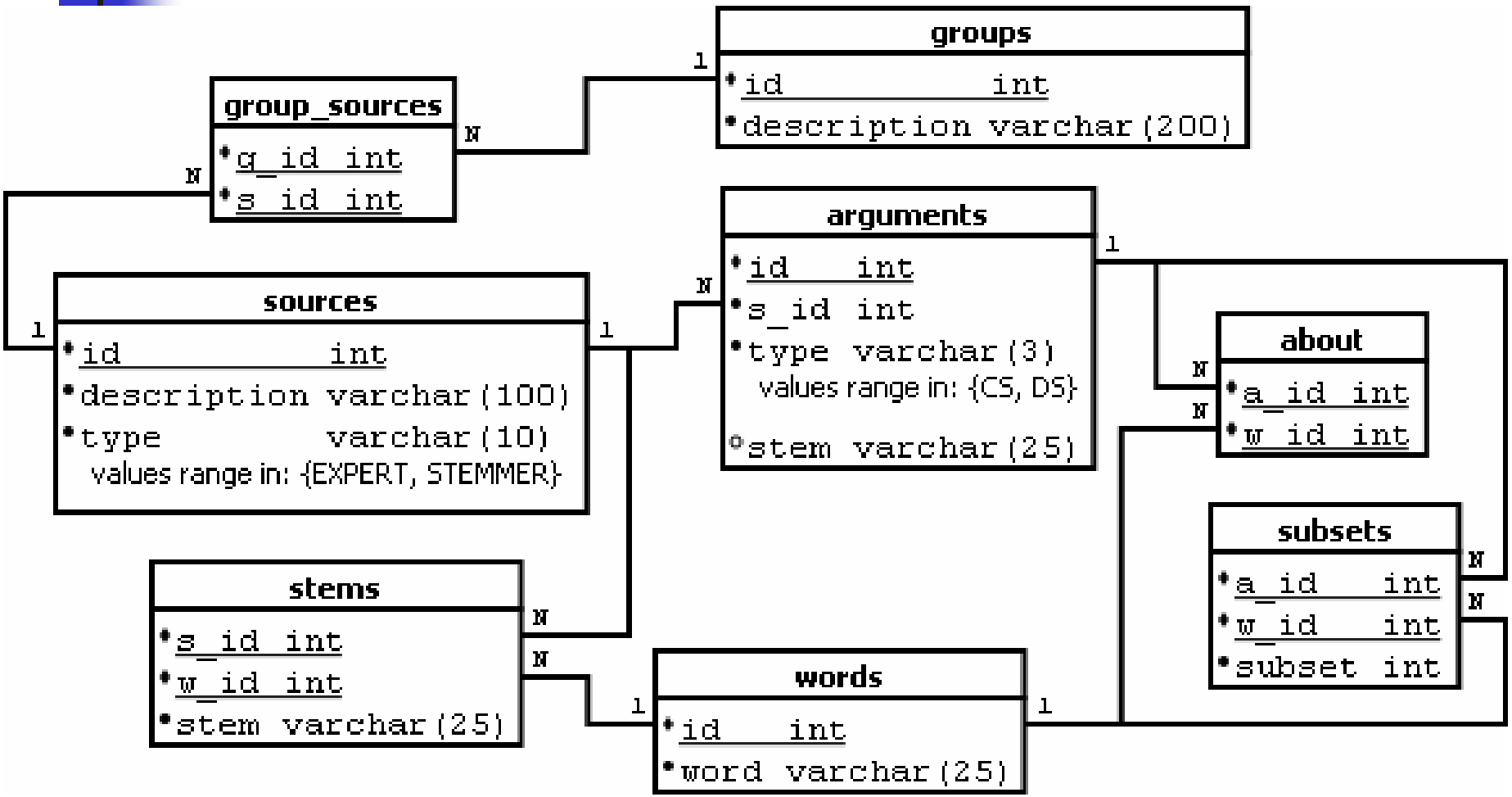
KUND:
exist in every
longest
most frequent



x in CS(x) – requirements

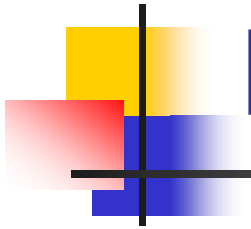
- Requirement to select a stem that exists in every word of the set. It comes from the need to get the stem with simple suffix removal (no replacements).
- Requirement to be the longest one. It comes from the need to not over-conflate (conflate with neighbour words which have other meanings).
- The requirement to be the most frequent. It is because it leaves fewer cases that impose adaptation of stemmer.

Database structure



Database
Codified expert's arguments

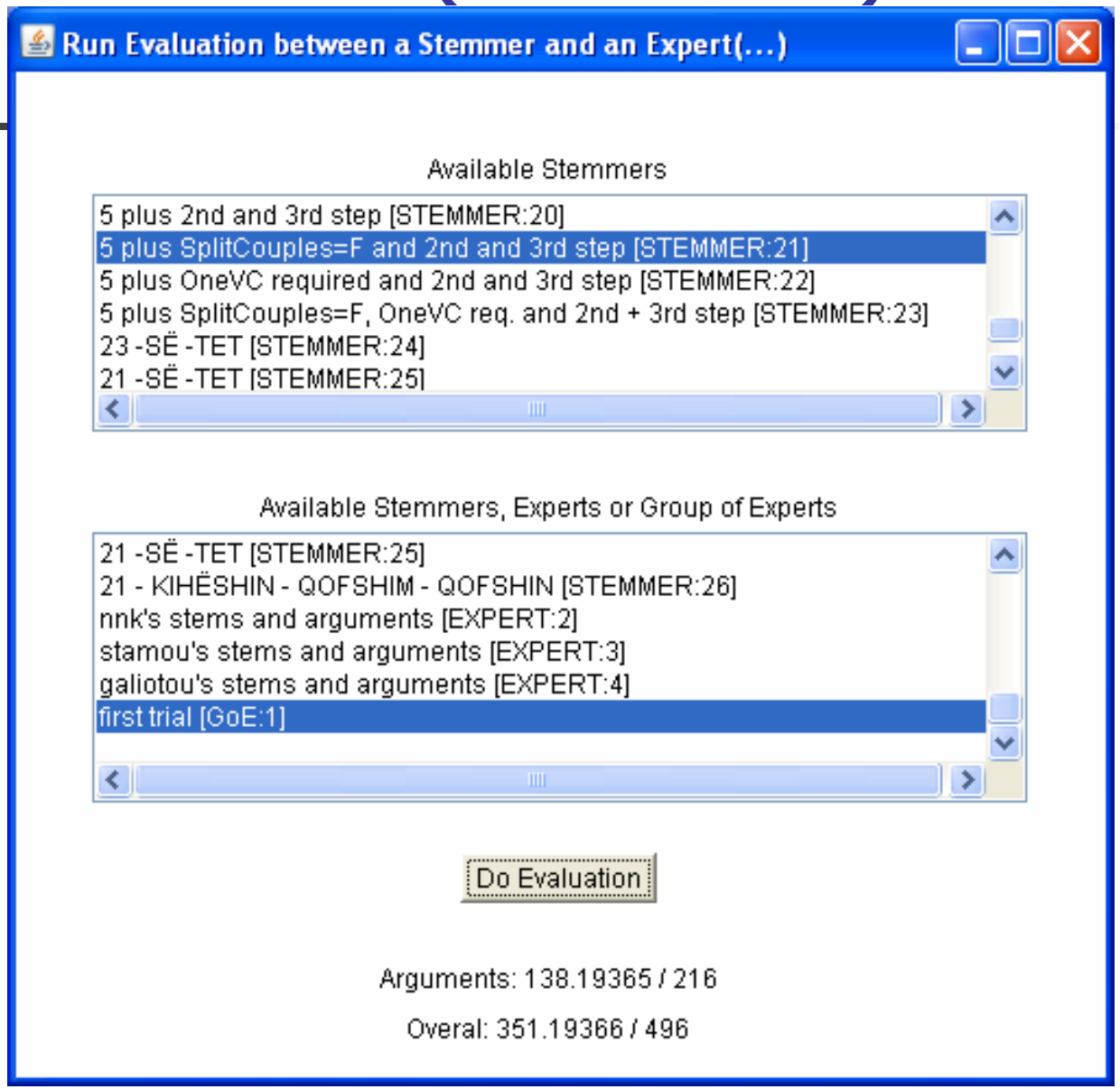
INSERT INTO **arguments** values (29, 3, 'CS', 'HIM');
INSERT INTO **about** values (29,1489);
INSERT INTO **about** values (29,1490);
INSERT INTO **about** values (29,1491);
INSERT INTO **about** values (29,1492);
INSERT INTO **about** values (29,1493);
INSERT INTO **arguments** values (32, 3, 'DS', null);
INSERT INTO **subsets** values (32,1963,1);
INSERT INTO **subsets** values (32,1964,2);
INSERT INTO **arguments** values (123, 4, 'DS', null);
INSERT INTO **subsets** values (123,3172,1);
INSERT INTO **subsets** values (123,3173,1);
INSERT INTO **subsets** values (123,3174,1);
INSERT INTO **subsets** values (123,3175,1);
INSERT INTO **subsets** values (123,3176,1);
INSERT INTO **subsets** values (123,3177,2);
INSERT INTO **subsets** values (123,3178,2);



Matching Algorithm

- **Intra subset uniformity (**how much uniform are the stemmer's results intra subsets**)**
- **Inter subsets unevenness (**how much unevenness are the stemmer's results inter subsets**)**
- **Factors combination (**relative contribution between previous factors**)**

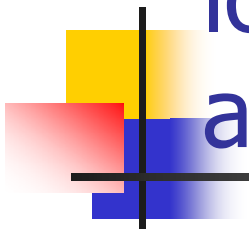
Interface for evaluating stemmers (Evaluator)



1st Builder: 1st step – Remove the longest suffix under 4 (optional) conditions

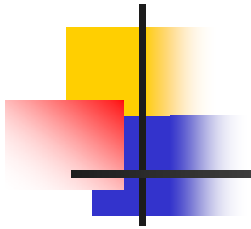
- *Active Suffix* condition (per suffix). Suffixes marked as inactive are not checked and consequently are not candidate for removal.
- *At Least Remain Letters* arithmetic parameter. A suffix removal is permitted only when the remaining word part contains a number of letters which is equal or greater than the parameter's value.
- *One VC* optional condition. If enabled, a suffix removal is permitted only when the remaining word part contains at least one VC pattern (where V is a sequence of one or more vowels and C is a sequence of one or more consonants). Otherwise, it doesn't matter if no one VC remains after suffix removal.
- *Split Couples* optional condition. If disabled, a suffix removal is permitted only when the last letter of the remaining word part followed by the first letter of the suffix being removed do not constitute a Couple. Otherwise, the suffix is removed without checking if a Couple is split.

1st Builder: 2nd step – Remove the longest suffix under 3 optional and 1 mandatory condition



- *Active Suffix* optional condition (per suffix).
- *At Least Remain Letters* optional arithmetic parameter.
- *Split Couples* optional condition.
- *VCVCVC* mandatory condition. The suffix removal is permitted only when the remaining word part contains at least the VCVCVC pattern.

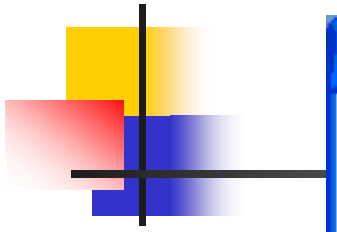
1st Builder: 3rd step – Remove ending consonants under 1 optional and 1 mandatory condition



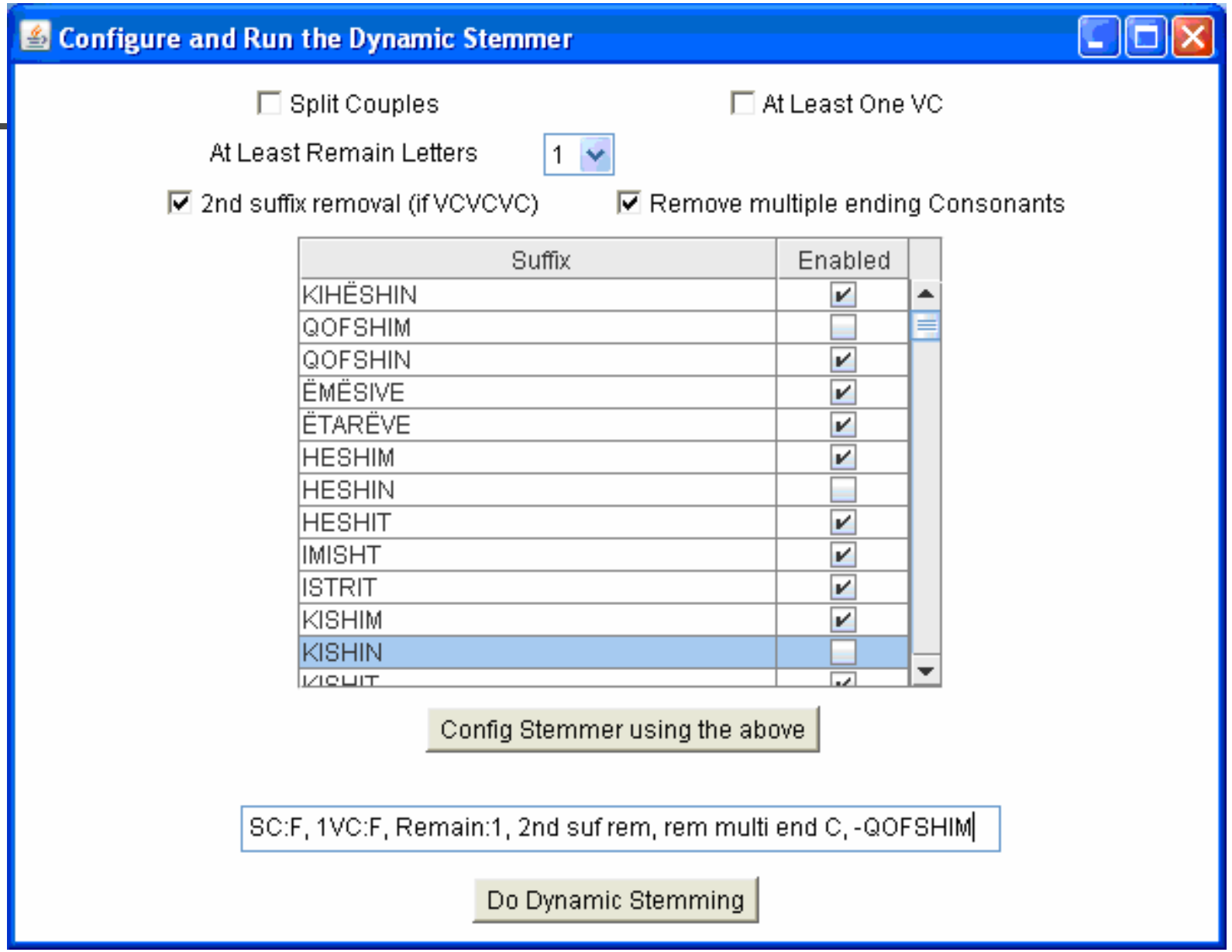
- *At Least Remain Letters* optional arithmetic parameter.
- *Do not Split Couples* mandatory condition. The ending consonant is removed if the previous letter is also a consonant and together they do not constitute a Couple.

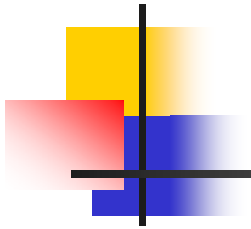
- Removal is repeated in case of multiple ending consonants.

1st Builder: Interface



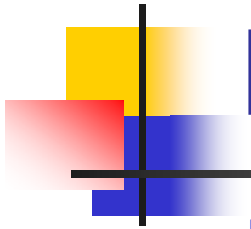
Nikitas N. Karanikolas – UNS – May 2017 – Building Stemmers for IR





1st Builder: Overview

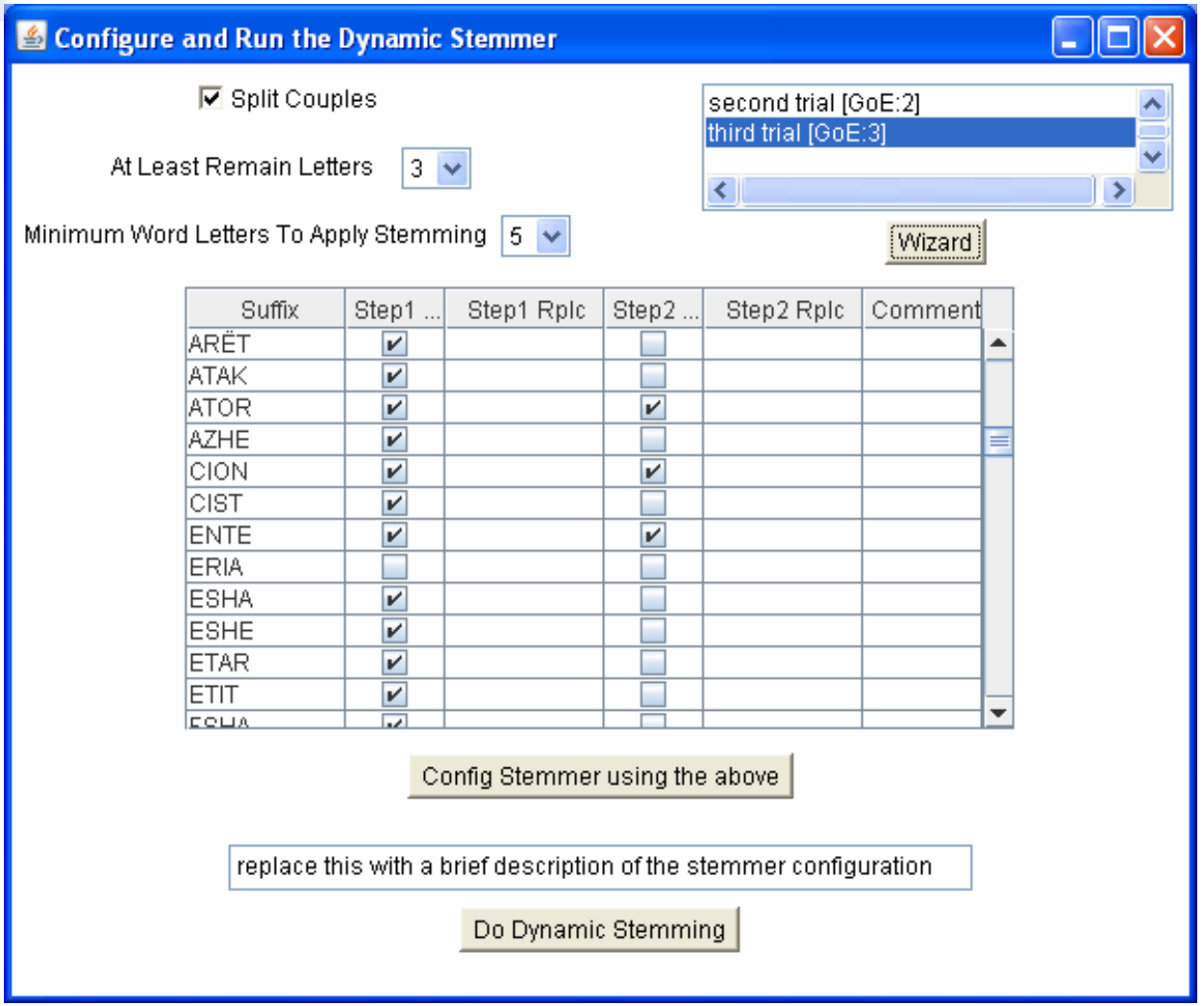
- So far, our approach for building stemmers was based in one set of suffixes that were used in both of the first two steps.
- The application of the second step was optional and this was one of the user's interventions in order to create/define alternative trial stemmers.
- Enabling the second step was guidance to a Paice like stemmer. There were also other available configuration options (split or do not split couples; number of remaining letters after suffix removal; etc) that the user could use in order to create/define alternative trial stemmers.
- There was also a third (optional) step for removing multiple ending consonants. The later was guidance to a Lovins like stemmer.
- However, the set of (selected by user) active suffixes was the same in both (first and second) steps, while the operation of the third step was not affected by the set of active suffixes.



Different suffixes for each step

- The available classical solutions for stemming words (e.g. Porter's) gave us another paradigm where the suffixes (endings) removed in each step are not same.
- Since many researchers still use the Porter's stemmer, we decided to adopt this paradigm and provide to the user the ability to enable/disable different suffixes for each step
- As we will see the builder uses a table with six columns. Columns two (*Step1*) and four (*Step2*) provide the user the abilities to:
  - disable Suffix (provided in the first column) in both steps;
  - disable Suffix in first step and enable it in the second step;
  - enable Suffix in first step and disable it in the second step;
  - enable Suffix in both steps.

2nd Builder: Interface



The idea behind the Builder's Wizard

Training Set

Ref.Num	Word	Stem	Argument

	Counters			
Suffix	1-PU	1-NU	2-PU	2-NU

- Only explicit CS and implicit (passive) CS arguments are considered. The DS/CS arguments are not considered.
- For each line of the examined arguments the algorithm tries to adapt the (primary) stemmer's result with the x of the CS(x) argument.
- For this adaptation some suffixes should enabled/disabled for the first or the second step. The relevant counter of each suffix participating in the adaptation is increased by one

2 examples of Wizard's operations (quadruples and relevant suffixes)

RefNum	Word	Stem	Argument
3562	PRONARET	PRONAR	PRON

Suffix	1-PU	1-NU	2-PU	2-NU
AR			+1	
ET	+1			

RefNum	Word	Stem	Argument
2015	KRYERA	KRYE	KRYER

Suffix	1-PU	1-NU	2-PU	2-NU
RA		+1		
A	+1			

Wizard's filters that activate / deactivate suffixes

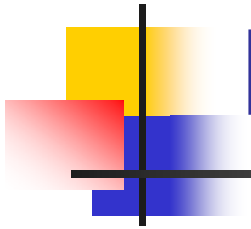
if (Step1-NU > Step1-PU)

Disable Suffix on 1st step

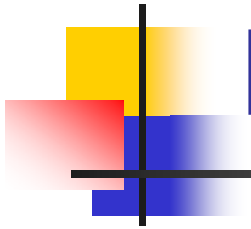
if (Step2-PU > Step2-NU)

Enable Suffix on 2nd step

Improvements of the Matching Algorithm



- To be explained in some next version



Evaluation – dimensions

- by 5000 distinct words
- 2100 quadruples of the form (<Ref.Num.>, <Word>, <Stem>, <Argument>)
- 470 stopwords (adj, prep, aux.verbs, etc)
- 380 suffixes
- 5 IR experts
- 4 Builder configurations

4 builder configurations Evaluation results for expert V

Wizard's configuration	Harmonization rates
V ¬SC RL:3 MWL:5	300.40 / 405 = 74.2%
V ¬SC RL:1 MWL:5	295.40 / 405 = 72.9%
V SC RL:3 MWL:5	298.65 / 405 = 73.7%
V SC RL:1 MWL:5	292.65 / 404 = 72.3%

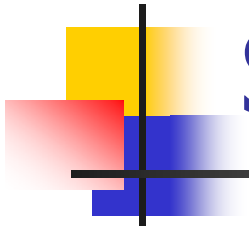
SC = split couples
RL = remain letters
MWL = minimum word letters to apply stemming

Best evaluation result per Expert

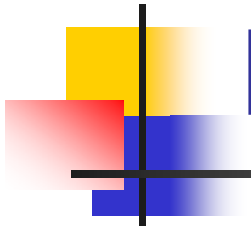
Expert	harmonization with primary stemmer	harmonization with best wizard's stemmer	Improvement
V	66,4%	74,2%	11,7%
F	66,1%	69,5%	5,1%
A	61,1%	69,8%	14,2%
S	69,8%	81,3%	16,5%
K	73,6%	80,7%	9,6%

Average improvement 11.4%

Evaluation Results – Best Stemmer configuration

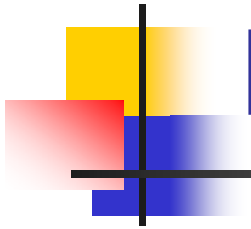


- VFASK ¬SC RL:3 MWL:5
- Overal improvement is 9.6% (74.1/67.6 = 1.096),
- Slightly less than the average improvement.
- This is an expected reduction since the more arguments there are, the more the conflicts there are by activation/deactivation of suffixes



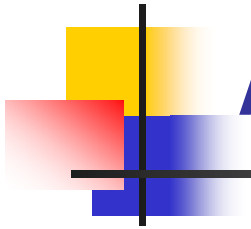
Polish Language

- The Polish language is a highly inflectional language.
- Verbs are inflected according to voice, tense, mood, gender, number and person.
- Greek language, verbs are inflected according to voice, tense, mood, number and person.
- Gender does not affect the formation of a verb in the Greek language.



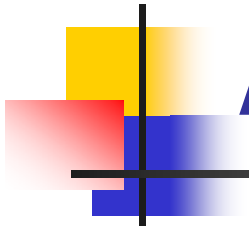
Existing stemmers

- Błażej Kubiński's stemmer is a rule based stemmer that remove endings (suffixes)
- Suffix removal is based on simple rules (defined by human experts').
- In some cases removes prefixes (in some adjectives).
- It is implemented in python programming language.
- The idea is based on Porter's Algorithm.
- This stemmer does not use replacements.



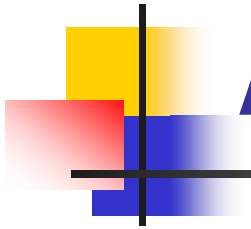
Another Polish stemmer

- Andrzej Białecki's Stempel stemmer [3] is another rule based stemmer
- It separates the basic algorithm from the data that adapt the execution flow.
- The basic algorithm is result of the Egothor project that developed a universal stemmer.
- Data are transformation rules (patch commands) defined separately for each language/stemmer.
- Transformation rules are not defined by human experts but they are learned from a training corpus and they are stored in data tables.
- Thus, the Stempel stemmer is a compilation of the Egothor universal stemmer with patterns extracted by learning from Polish corpuses.



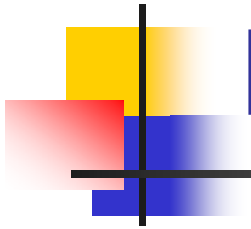
Another one Polish Stemmer

- Dawid Weiss's Lametyzator is a dictionary-based stemmer.
- Internally, Lametyzator uses pairs (inflected form – stem).
- The data (pairs) of Lametyzator comes from another project (Polish dictionary for ispell).
- An efficient representation of a huge number of such pairs is based in a finite state automaton.
- Thus, for any word (inflected form) that Lametyzator has in its database, the corresponding stem can be returned.



About Polish language

- Polish language has 32 letters and 7 digraphs.
- Each Polish digraph corresponds to a single sound and actually to a single consonant (digraph consonants).
- We use the term couples for digraph vowels, digraph consonants, and diphthongs.



Feet for Polish

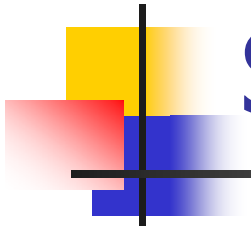
- All features of the methodology can apply for Polish.
- In the beginning a list of available for the target language suffixes should be imported in our system (methodology).
- These suffixes can be extracted from grammar books of the target language by target language speakers having high school knowledge of language (not IR experts).
- An external primary single-step stemmer that simply removes the longest matching suffix can also be easily programmed.
- IR experts can declare their arguments against the primary stemmer's results.
- The Wizard can facilitate the user (IR expert) to create easily good trial stemmers.
- Configuration options ("At least remains letters", "Minimum word letters to apply stemming", "Split couples") can be used for fine tuning trial stemmers.
- The Harmonization measurement function is built in the system.

CS argument for Polish

Ref. No	Word	Stem	Explanation	Argument
1118	głębiej	głęb	βαθύ	CS(głęb)
1119	głęboką	głębok	βαθιά	
1120	głębokie	głębok	βαθιά	
1121	głęboko	głębok	βαθιά	
1122	głębszym	głęb	βαθύτερο	

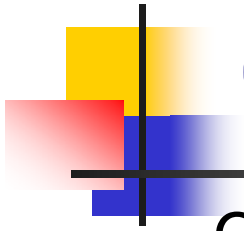
DS/CS argument for Polish

Ref. No	Word	Stem	Explanation	Argument	
1289	Idealne	ide	τέλειο	DS	CS_1
1290	Idealnego	ide	τέλειου		
1291	Idealnych	ide	τέλειους		
1292	Ideał	ide	τελειότητα		
1293	Ideą	ide	ιδέα		CS_2
1294	Idee	ide	ιδέα		
1295	Ideę	ide	ιδέα		
1296	ideologii	ideolog	ιδεολογία		CS_3



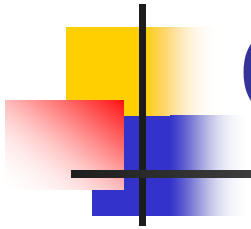
Split couples

- Our system offers the “split couples” as a configurable parameter.
- This did not be useful in previous experiments (Albanian language). Albanian language do not permit splitting digraphs during the application of inflectional rules that produce words.
- However, the Polish language justified our choice. The Polish words “koszony” (participle, passive, present perfect, male, singular, nominative) and “kosić” (verb, active, present, subjunctive, singular, second person) are inflected forms of verb barber (to cut someone’s hairs).
- The first one contains the digraph “sz” while the second one has only the letter “s”.



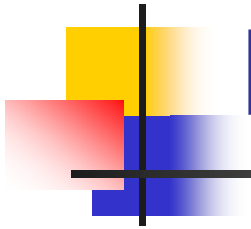
Couples

- Couples (consonant digraphs) in Albanian:
  dh gj ll nj rr sh th xh zh
  - dh → Greek δ
  - nj → Serbian Њ
  - th → Greek θ
  - xh → Serbian Ћ or Џ
- Couples (consonant digraphs) in Polish:
  ch cz dz dź dż rz sz
- Couples (vowel digraphs and diphthongs) in Greek:
  ει οι αυ ευ αη ...
  - εί → κλείνω
  - οι → άνθρωποι
  - ευ → εύλογο
  - αη → αηδόνι



Conclusions

- It seems that our methodology offers the proper facilities for building stemmers for the Polish language.
- Without having advanced knowledge of the Polish language.
- We need only basic knowledge of the Polish language:
  - alphabet,
  - Couples (digraphs),
  - a list of suffixes,
  - few documents.
- and volunteers to translate some Polish words to the language that IR experts speak.



Future work

- extend the Wizard in order to also consider the DS/CS arguments
- An internal to the system Alphabet Reduction should be very interesting. In such case not any accent or diacritics removals should be conducted outside the system (before data insertion).
- to build a stemmer for Polish using our system but now with a big set of words (more that 5000 words) and compare the resulting stemmer with another existing rule based stemmer for Polish.

Our previous work in this domain

- **Nikitas N. Karanikolas**, Bootstrapping the Albanian Information Retrieval, 4th Balkan Conference in Informatics, September 17-19, 2009, Thessalonica, Greece, IEEE Computer Society's Conference Publishing Services and IEEE Xplore
- **Nikitas N. Karanikolas**, A methodology for building simple but robust stemmers without language knowledge: Overview, data model and ranking algorithm. CompSysTech'2013: 14th International Conference on Computer Systems and Technologies, June 2013, Ruse, Bulgaria. ACM ICPS, doi:10.1145/2516775.2516783
- **Nikitas N. Karanikolas**, A methodology for building simple but robust stemmers without language knowledge: Stemmer configuration. Procedia, Social and Behavioral Sciences, vol. 147, pp. 370-375, 2014, doi:10.1016/j.sbspro.2014.07.113
- **Nikitas N. Karanikolas**, Supervised learning for building stemmers. Journal of Information Science, Vol. 41 (3), pp. 315-328, 2015, doi:10.1177/0165551515572528
- **Nikitas N. Karanikolas**, "Building Stemmers for the Polish Language". PCI 2016, November 10 - 12, 2016, Patras, Greece

Building stemmers for Information Retrieval …

- Thank you for your attention,
- I will try to answer Questions.

Building stemmers for Information Retrieval …

- **AUTHOR:**
- Nikitas N. Karanikolas,
- Professor,
- Dept. of Informatics,
- Technological Educational Institute (TEI) of Athens,
- http://users.teiath.gr/nnk/
- nnk@teiath.gr