Discovering Textual Structures: Generative Grammar Induction using Template Trees

Thomas Winters, Luc De Raedt
KU Leuven, Belgium

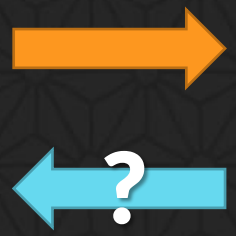
@thomas_wint
firstname.lastname@cs.kuleuven.be

Motivation

Generative model designers often
write generative grammars by hand.

Can we bootstrap this by **providing some exemplars**?

S → **<Hero>** is going to **<Action>**
Hero → **<Name>** the **<Job>**
Name → Alice | Bob | Cathy
Job → hunter | knight | cook
Action → fish | slay a dragon | relax



Alice the hunter is going to fish
Cathy the cook is going to slay a dragon
Bob the cook is going to fish
Alice the knight is going to relax

Motivation

Automatically **inducing generative context-free grammars** from text with latent templates

Problem:

Most grammar induction algorithms don't *"acknowledge"* templates, but prefer **part-of-speech**-like structures

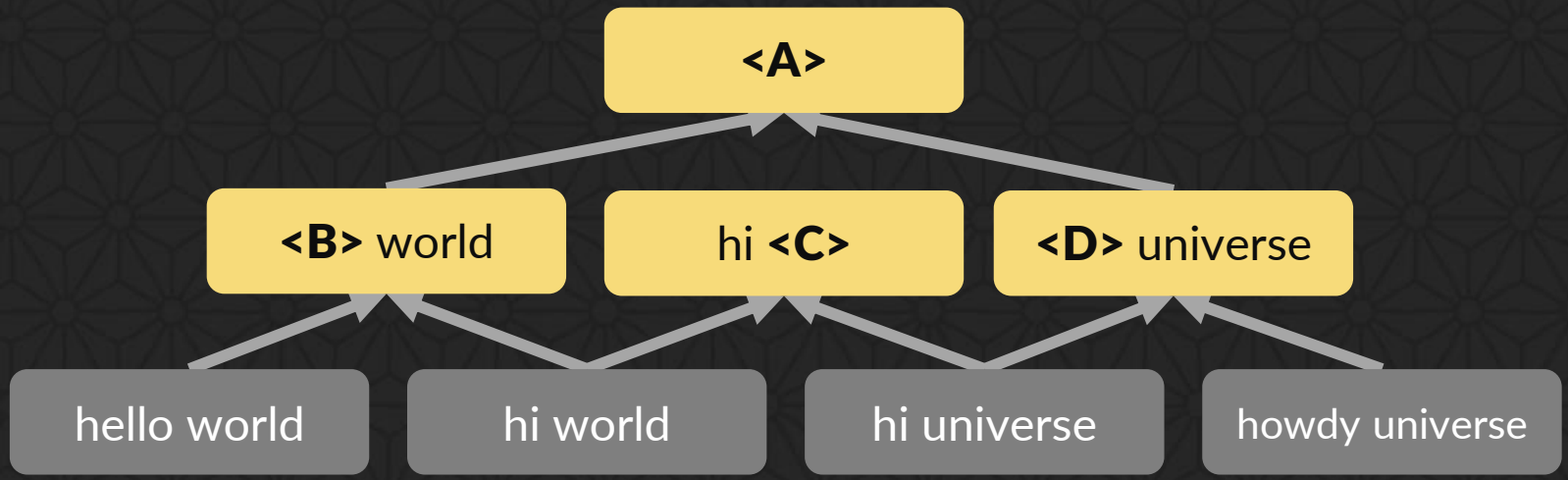
Generative grammars often **use template-like structures**!

Solution:

Introducing notion of "**Template Trees**"

Template Tree

= connected acyclic directed graph where each node represents a template that is more general than the template of all its child nodes



Created by **iteratively merging closest** templates

GITTA

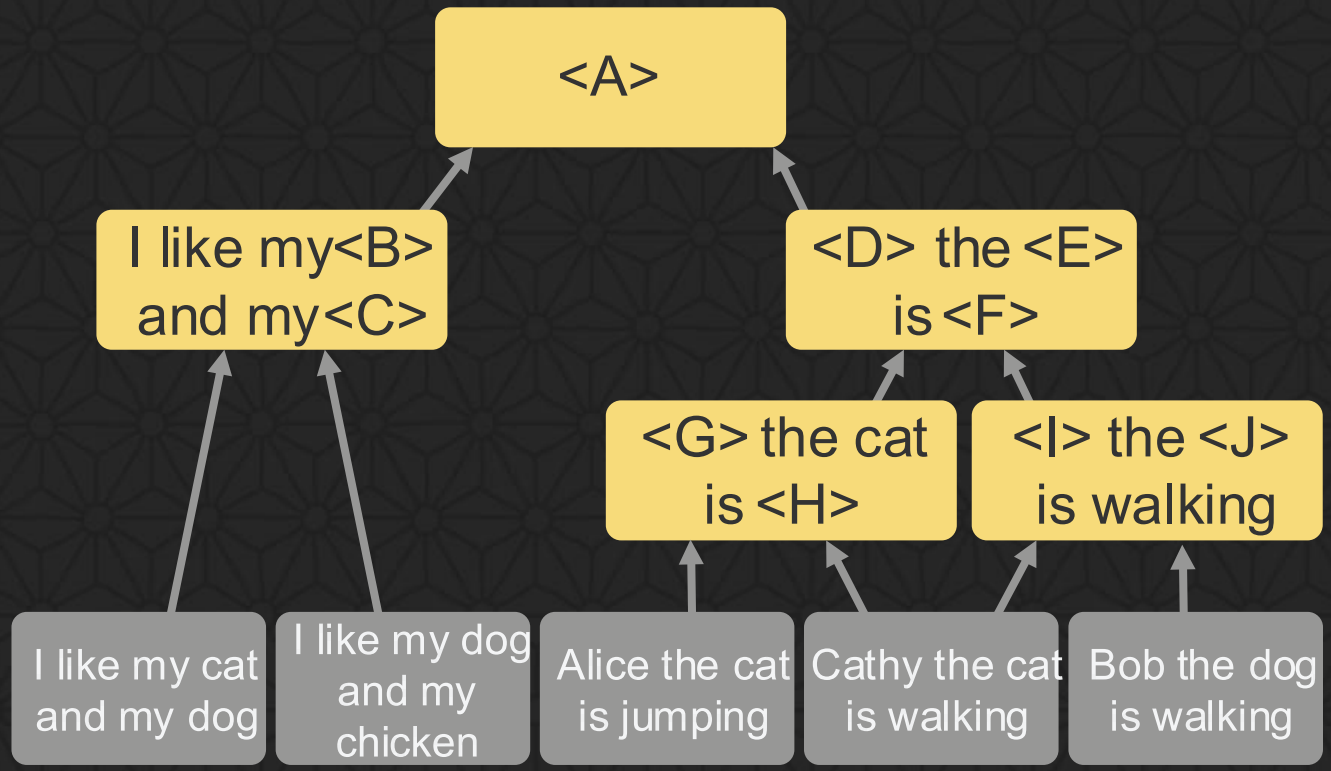
Grammar Induction using a Template Tree Approach

1. Creates **an initial template tree**
2. **Prunes redundant children** for each node *(all descendent leaves covered by other children)*
3. **Merges slots** *(when values have similar value sets)*
4. **Simplifies tree** using slot content
5. Step 3 & 4 until **convergence**

Example input

1. "I like my cat and my dog"
2. "I like my dog and my chicken"
3. "Alice the cat is jumping"
4. "Bob the dog is walking"
5. “Cathy the cat is walking”

Join **closest strings iteratively** into **templates with slots** to create **Template Tree**



A: I like my <B> and my <C>
| <D> the <E> is <F>

B: cat | dog

C: chicken | dog

D: <G> | <I>

E: <H> | cat

F: walking | <J>

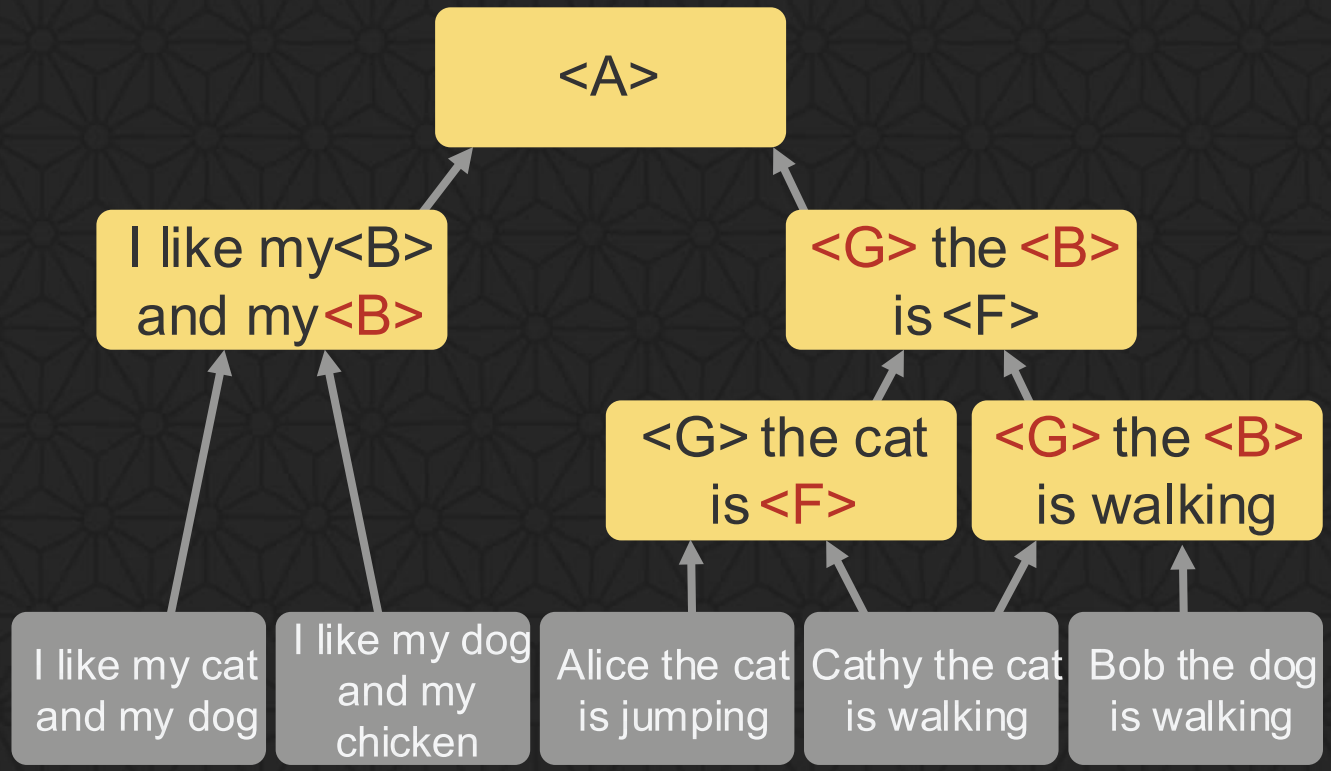
G: Bob | Cathy

H: cat | dog

I: Alice | Cathy

J: walking | jumping

Merge similar slots and rename slots in **Template Tree**



A: I like my <B> and my <B>
| <G> the <B> is <F>

B: chicken | cat | dog

C: <B>

D: <G>

E: <B>

F: walking | jumping

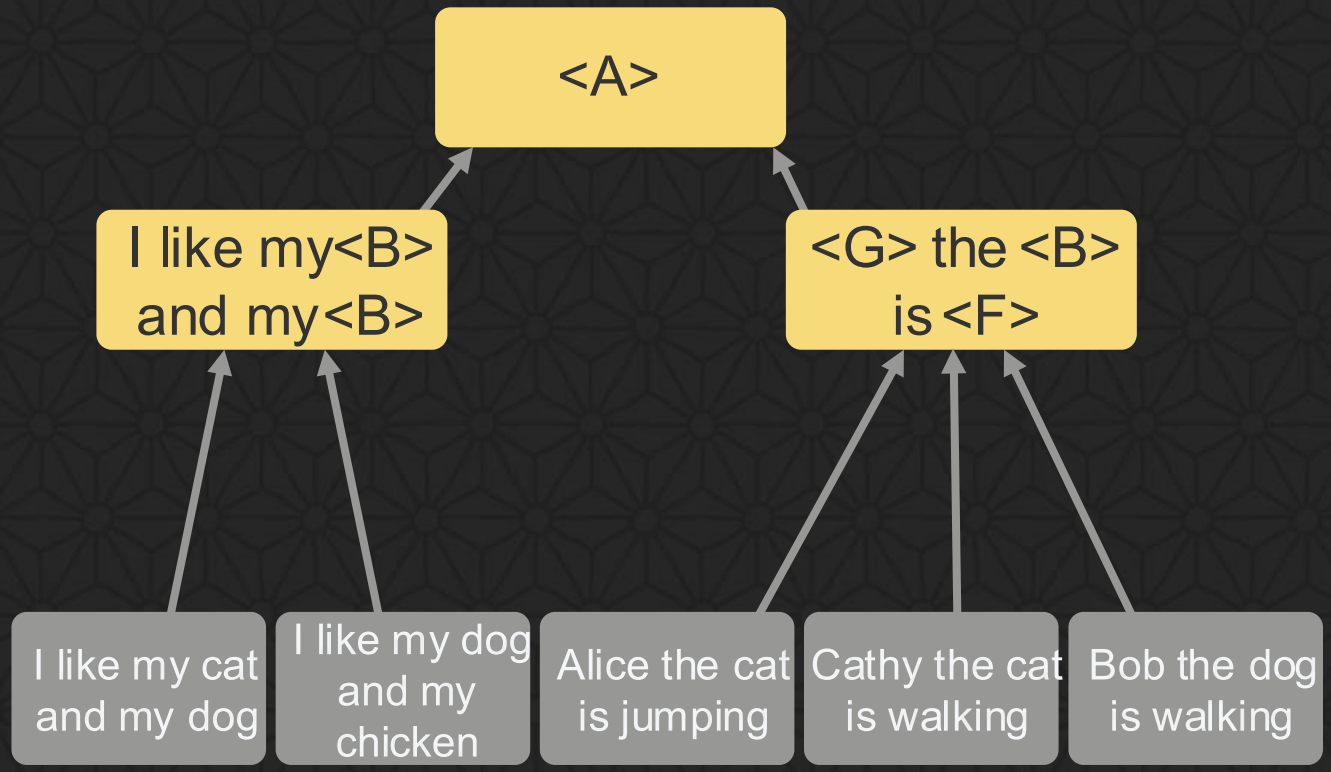
G: Alice | Bob | Cathy

H: <B>

I: <G>

J: <F>

Iteratively **merge and simplify tree** & recalculate templates **until convergence**



A: I like my <B> and my <B>
| <G> the <B> is <F>

B: chicken | cat | dog

F: walking | jumping

G: Alice | Bob | Cathy

Output

S → I like my <B> and my <B>
| <G> the <B> is <F>

B → chicken | cat | dog

F → walking | jumping

G → Alice | Bob | Cathy

Evaluation: Reverse-engineering Twitterbots

Given 25/50/100 example generations of Tracery grammar, try to reconstruct grammar

Grammar G				I from 25 examples			I from 50 examples			I from 100 examples		
id	*Name*	$\#L_G$	$\#R_G$	$\in L_G$	$\notin L_G$	$\#R_I$	$\in L_G$	$\notin L_G$	$\#R_I$	$\in L_G$	$\notin L_G$	$\#R_I$
1	botdoesnot	380292	363	648	0	64	2420	0	115	1596	4	179
2	BotSpill	43452	249	75	0	32	150	0	62	324	0	126
3	coldteabot	448	24	39	0	38	149	19	63	388	9	78
4	hometapingkills	4080	138	440	0	48	1184	3240	76	2536	7481	106
5	InstallingJava	390096	95	437	230	72	2019	1910	146	1156	3399	228
6	pumpkinspiceit	6781	6885	25	0	26	50	0	54	100	8	110
7	SkoolDetention	224	35	132	0	31	210	29	41	224	29	49
8	soundesignquery	15360	168	256	179	52	76	2	83	217	94	152
9	whatkilledme	4192	132	418	0	45	1178	0	74	2646	0	108
10	Whinge_Bot	450805	870	3092	6	80	16300	748	131	59210	1710	222

Table 1: Grammar induction results given a specific number of random generations of G, measuring median number of generations of the induced grammar I that are in and not in the target language, as well as their median sizes, over five runs.

GITTA finds **decent generalisations**, but difficulties with **subsequent slots**, especially if slots have large variations in number of words for their slots values.

Conclusion

GITTA is a promising approach for **learning generative grammars** from texts with **latent templates**

Code: https://github.com/twinters/gitta