Solving Hofstadter's analogies using Structural Information Theory

Geerten Rijsdijk (geerten.rijsdijk@student.uva.nl) and Giovanni Sileno (g.sileno@uva.nl), **University of Amsterdam**

BACKGROUND & MOTIVATION

Analogies are common part of human life; our ability to handle them is critical in problem solving, humor, metaphors and argumentation. This paper introduces a method to solve string analogies based on a hybrid inferential process integrating *Structural Information Theory* (SIT) with a metric-based processing. Results are discussed against two empirical experiments. The work comes together with the development of a Python version of the SIT encoding algorithm PISA, adequately extended for our purposes

Hofstadter's Analogies: A proportional analogy is expressed as '**A** is to **B** what **C** is to **D**'. In order to study analogies, Douglas Hofstadter proposed working with analogies in a simple domain, in which elements consist of strings of letters:

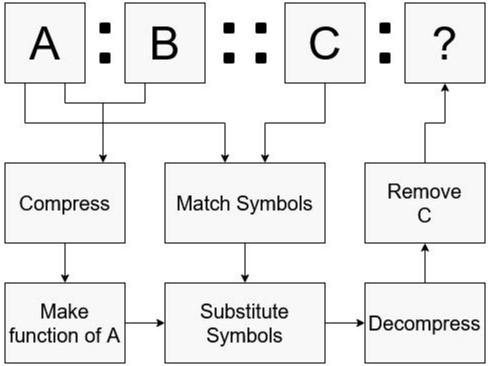
ABC : ABD :: IJK : ?

To solve such analogies, Hofstadter developed the algorithm *Copycat*, which was later extended in the program *Metacat.*

Structural Information Theory is a theory on perceptual organization based on symbolic processing. The SIT coding language is based upon the following operators:

- *Iteration* **3*(A)** ⇒ **AAA**
- *Symmetry* **S[(A)(B),(C)]** ⇒ **ABCBA**
- *Alternation* **<(A)>/<(B)(C)>** ⇒ **ABAC**

Encoding a string using these operators can greatly reduce the length of the description of the string ⇒ it is a form of *compression*.



METHOD

- **Compress:** An extended version of the PISA algorithm is used to create an encoding of the concatenation A+B. **<(AB)>/<(C)(D)>**
- **Make function of A:** Symbols that occur in B, but not in A, are replaced using symbols in A and distances from some of those, selected with some strategy: **<(AB)>/<(C)($+1)>**
- **Match Symbols:** A mapping between symbols in A and C is created. (It does not have to be one to one). **A:I, B:J, C:K**.
- **Substitute Symbols:** The symbol mapping is used to place new symbols into the SIT code. **<(IJ)>/<(K)($+1)>** ⇒ **<(IJ)>/<(K)(L)>**
- **Decompress:** A new symbol string is created by decompressing the code, i.e. using the SIT rules to work out the operators. **IJKIJL**
- **Remove C:** Finally, string C is removed from the new symbol string. If C does not occur in this new code, the answer is discarded. **IJL**

EXPERIMENTS & RESULTS

Answers generated by the solver were compared to human answers and to the results given by Metacat. The first dataset is by *Murena et. al.* (2017). All questions are of form **ABC:ABD::*X*:?**

Given X	Solutions	Selected by	P_s	P_M	Given X	Solutions	Selected by	P_s	P_M
IJK	IJL	**93%**	**1**	**1**	BCD	BCE	**81%**	2	**1**
	IJD	2.9%	-	-		BDE	5.9%	**1**	-
BCA	BCB	**49%**	3	2	IJJKKK	IJJLLL	**40%**	**1**	2
	BDA	43%	**1**	**1**		IJJKKL	25%	2	**1**
AABABC	AABABD	**74%**	**1**	**1**	XYZ	XYA	**85%**	**1**	-
	AACABD	12%	-	-		IJD	4.4%	-	-
IJKLM	IJKLN	**62%**	**1**	**1**	RSSTTT	RSSUUU	**41%**	**1**	**1**
	IJLLM	15%	-	-		RSSTTU	31%	2	-
KJI	KJJ	**37%**	**1**	**1**	MRRJJJ	MRRJJK	**28%**	2	**1**
	LJI	32%	-	2		MRRKKK	19%	**1**	2
ACE	ACF	**63%**	**1**	**1**					
	ACG	8.9%	-	-					

Results: generating given human answers: our solver 16/22 (73%), Metacat 14/22 (64%); most common human answer in the top 2 generated answers: 10/11 (91%) both; best answer matching most common human answer: 8/11 (73%) both.

The second dataset was created by us, about 20 more complex questions. The problems were answered by 35 participants.

Given problem	Solutions	Selected by	P_s	P_M
ABA:ACA::ADA:?	AEA	**97.1%**	**1**	**1**
	AFA	2.9%	-	-
ABAC:ADAE::BACA:?	DAEA	**60%**	2	-
	BCCC	28.6%	21	-
AE:BD::CC:?	DB	**68.5%**	3	**1**
	CC	17.1%	-	2
ABBB:AAAB::IIIJJ:?	IIJJJ	**57.1%**	**1**	-
	JJIII	14,3%	-	-
ABC:CBA::MLKJI:?	IJKLM	**88.6%**	**1**	**1**
	-	-	-	-
ABCB:ABCB::Q:?	Q	**100.0%**	**1**	-
	-	-	-	-
ABC:BAC::IJKL:?	JIKL	**54.3%**	-	-
	KIJL	14.3%	2	-
ABACA:BC::BACAD:?	AA	**57.1%**	**1**	-
	BCD	31.4%	-	-
AB:ABC::IJKL:?	IJKLM	**85.7%**	**1**	**1**
	IJKLMN	11.4%	-	-
ABC:ABBACCC::FED:?	FEEFDDD	**91.4%**	2	1
	-	-	-	-

ABC:BBC::IKM:?	JKM	**57.1%**	7	-
	KKM	37.1%	2	-
ABAC:ACAB::DEFG:?	DGFE	**68.6%**	2	-
	FGDE	14.3%	1	-
ABC:ABD::CBA:?	DBA	**51.4%**	1	2
	CBB	45.7%	2	**1**
ABAC:ADAE::FBFC:?	FDFE	**94.3%**	1	-
	FDFA	2.9%	-	-
ABCD:CDAB::IJKLMN:?	LMNIJK	**80.0%**	-	-
	-	-	-	-
ABC:AAABBBCCC::ABCD:?	AAABBBCCCDDD	**74.3%**	1	**1**
	AAAABBBBCCCCDDDD	17.1%	-	-
ABC:ABBCCC::ABCD:?	ABBCCCDDDD	**85.7%**	-	-
	ABBCCCDDD	8.6%	1	-
ABBCCC:DDDEEF::AAABBC:?	DEEFFF	**77.1%**	1	-
	DCCDDF	8.6%	-	-
A:AA::AAA:?	AAAAAA	**62.8%**	1	-
	AAAA	25.7%	2	**1**
ABBA:BAAB::IJKL:?	JILK	**71.4%**	-	-
	JIJM	11.4%	5	-

Results: generating given human answers: our solver 16/20 (80%), Metacat 7/20 (35%); most common human answer in the top 2 generated answers: 13/20 (65%) vs 8/20 (40%); best answer matching most common human answer: 10/20 (50%) vs 6/20 (30%).

Besides providing a better alignment to human answers, our solver is much ***faster***. Metacat sometimes took over 10 minutes to generate a single answer, whereas our solver generally a couple of seconds.