

UC San Diego



Petuum

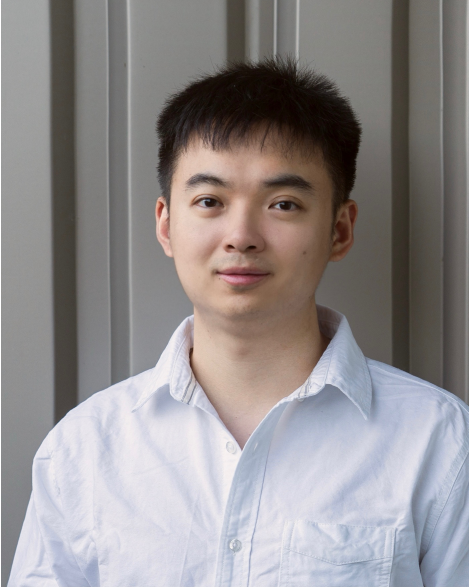
A “Standard Model” for Machine Learning

Zhiting Hu, Hao Zhang, Eric Xing

DeepLearn 2023 Winter



Presenters



Zhiting Hu

Assistant Professor
@UCSD



Hao Zhang

Assistant Professor
@UCSD

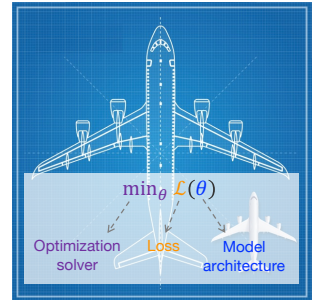


Eric P. Xing

Professor @ CMU
President @ MBZUAI
Co-founder @ Petuum

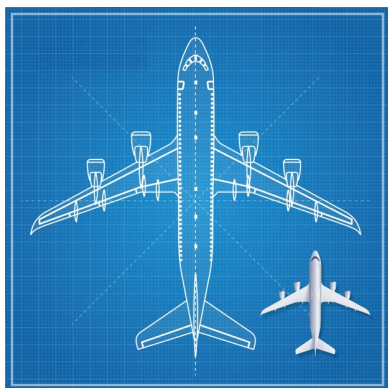
Schedule

- Lecture#1:** Theory: The Standard Model of ML
 A blueprint of ML paradigms for ALL experience
(Jan 19 Thursday, 4:45pm-6:15pm UK Time)
- Lecture#2:** Tooling: Operationalizing The Standard Model
 Compose your ML solutions like playing Lego
(Jan 20 Thursday, 1:00pm-2:30pm)
- Lecture#3:** Computing: Modern infrastructure for productive ML
 Automatic tuning, distributing, and scheduling
(Jan 20 Thursday, 4:45pm-6:15pm)



Tooling:

Operationalizing The “Standard Model”



$$\min_{q, \theta} -E + D - H$$



Recap of Part-I: Standard Equation

$$\min_{q, \theta} - \mathbb{E}_{q(x, y)} \left[f(x, y) \right] + \beta \mathbb{D} \left(q(x, y), p_{\theta}(x, y) \right) - \alpha \mathbb{H}(q)$$

3 terms:

Experience

- data examples
- rules
- reward
- adversarial models
- ...

Divergence

- Cross Entropy
- JS Divergence
- f-Divergence
- Wasserstein Dist.
- ...

Uncertainty

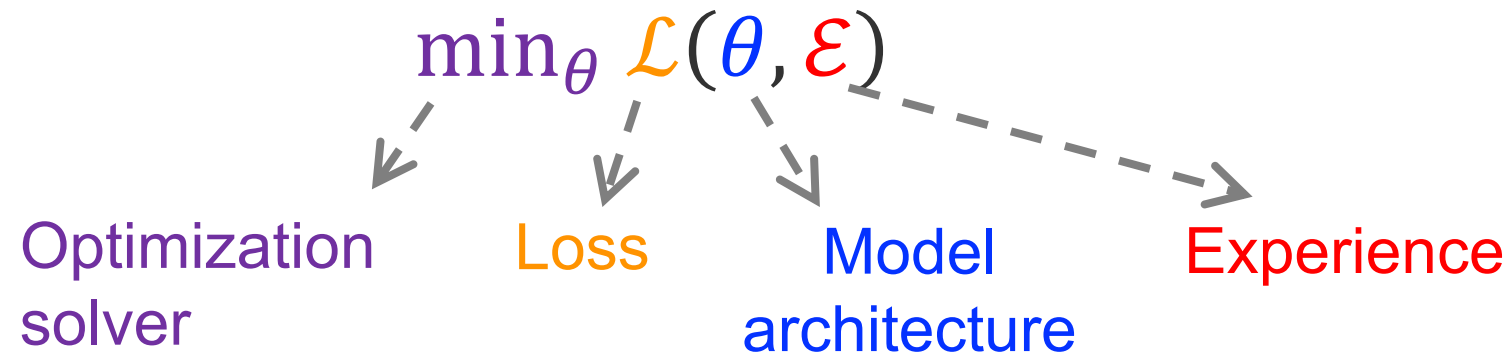
e.g., Shannon entropy

$$\min_{q, \theta} - \mathbb{E} + \mathbb{D} - \mathbb{H}$$



Recap of Part-I: Standard Equation

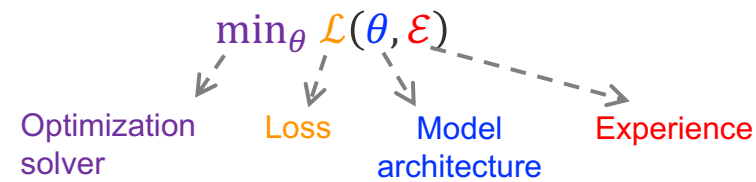
$$\min_{q, \theta} -\mathbb{E} + \mathbb{D} - \mathbb{H}$$



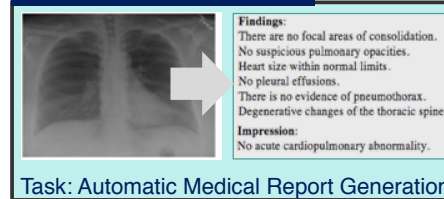
Part-II: Operationalizing The “Standard Model”

- ML solution design
 - Plugging arbitrary experiences in learning
- Tooling for composable ML

$$\min_{q, \theta} -E + D - H$$

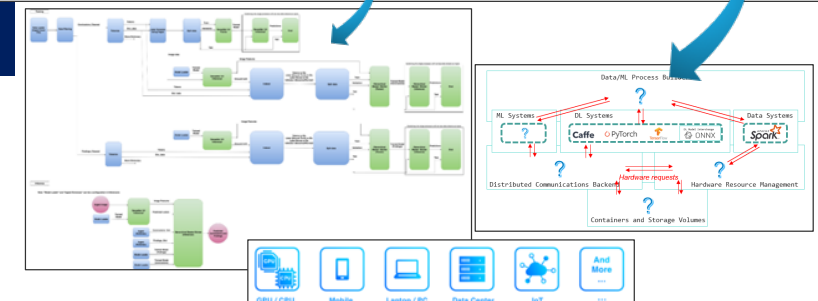
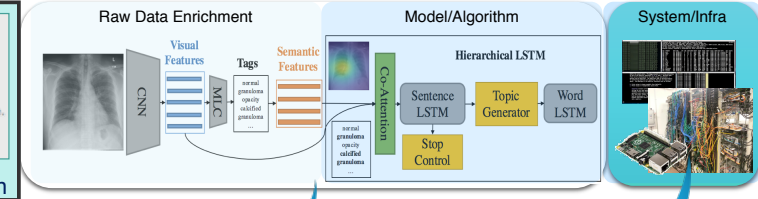
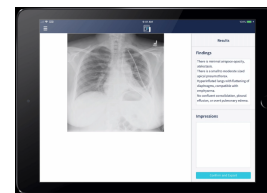


Extremely complex



Requires inter-operation between diverse systems

User interface for Doctors



Problem solving by plugging arbitrary available experiences in learning

Standard equation

$$\min_{q, \theta} - \mathbb{E}_{q(x, y)} \left[f(x, y) \right] + \alpha \mathbb{D} \left(q(x, y), p_{\theta}(x, y) \right) - \beta \mathbb{H}(q)$$

↓

$$f = w_1 \cdot f(x | \text{database}) + w_2 \cdot f(x | \text{book}) + w_3 \cdot f(x | \text{money}) + w_4 \cdot f(x | \text{person}) + \dots$$

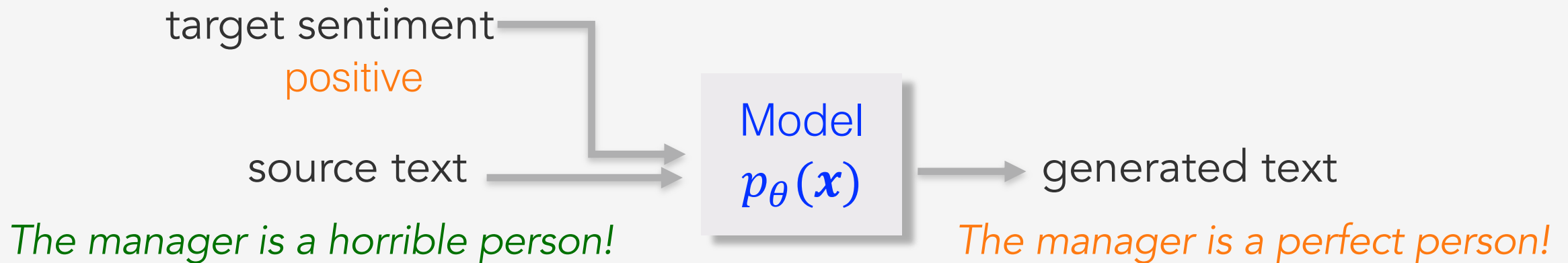


Problem: Controllable Text Generation

Ex. controlling sentiment

Goals: generating a new sentence that has the target sentiment while preserving all other aspects

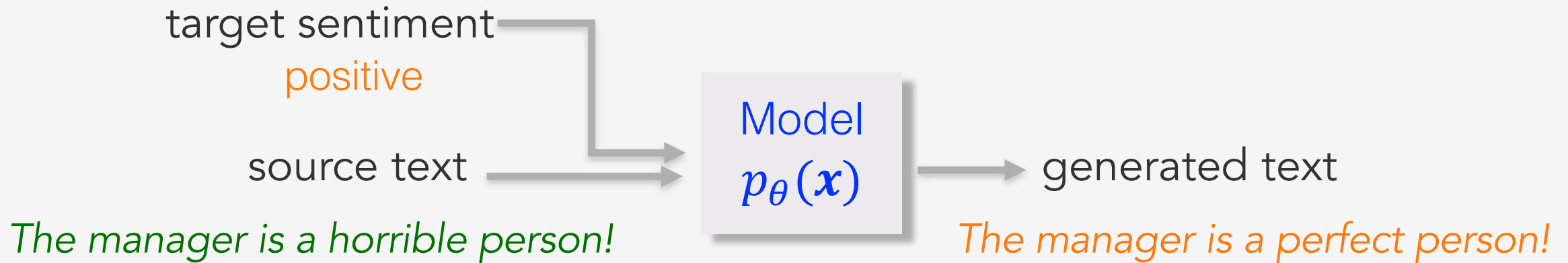
Key challenge: no direct supervision data



Designing the Solution

Goals: generating a new sentence that has the target sentiment while preserving all other aspects

1) Consider what experiences to use



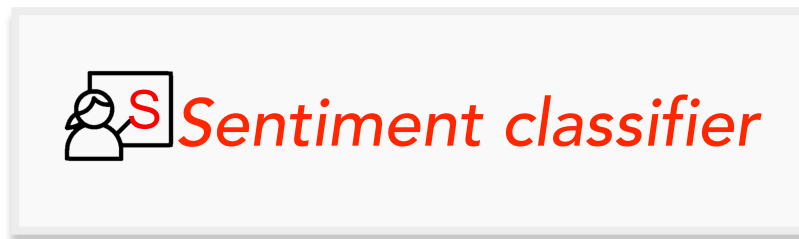
Designing the Solution

Goals: generating a new sentence that has the target sentiment while preserving all other aspects

1) Consider what experiences to use

 **Sentiment classifier**

The manager is a perfect person!



logit of target sentiment

Designing the Solution

Goals: generating a new sentence that has the target sentiment while preserving all other aspects

1) Consider what experiences to use

 *Sentiment classifier*

Designing the Solution

Goals: generating a new sentence that has the target sentiment while preserving all other aspects

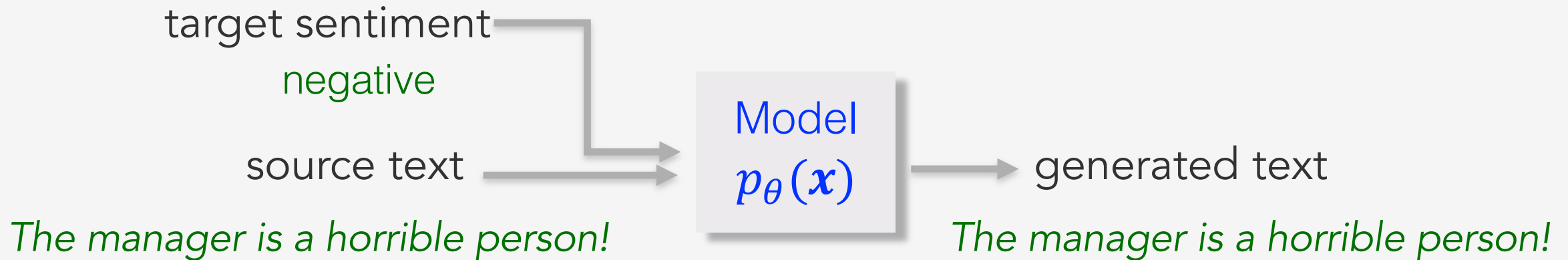
1) Consider what experiences to use



Sentiment classifier



**Self-construction
data examples**



Designing the Solution

Goals: generating a new sentence that has the target sentiment while preserving all other aspects

1) Consider what experiences to use



The manager is a perfect person!



Language model



perplexity score



Designing the Solution

1) Consider what experiences to use

 *Sentiment classifier*
  *Self-construction data examples*
  *Language model* ...

2) Plug experiences into the algorithm

Designing the Solution

1) Consider what experiences to use

 *Sentiment classifier*
  *Self-construction data examples*
  *Language model* ...

2) Plug experiences into the algorithm

Source text: The manager is a horrible person!

Experiences plugged in:

$$f = f(x | \text{Person S})$$

$$+ f(x | \text{Database})$$

$$+ f(x | \text{Person L})$$

Resulting generated text:

good good good good ...

The manager is a delicious person!

The manager is a perfect person!

[Hu et al., ICML'17]



Experimental Results

	Accuracy (↑)	Perservation (↑)	Language quality (↓)
Sentiment classifier 	98.9 😊	0.1	326.1
+ Self-construction data examples 	87.7 😊	65.6 😊	115.6
+ Language model 	91.2 😊	57.8 😊	47.2 😊

Applications in other controllable generation problems

Controlling sentiment

Pos The film is *full of imagination!*



Neg The film is *strictly routine!*

[Hu et al., ICML'17]

Rewriting content

He scored *23* points and pulled down *8* rebounds .

Name	LeBron James
Points	32
Rebounds	4
Assists	7

LeBron James scored *32* points, pulled down

4 rebounds, *and added 7 assists.*

Guiding conversation flow



Hi, how are you today?



Fine. Just finished *riding* along the river.



Cool! You can ride *bikes*, listen to music there too.



Yes. I like *Taylor Swift*.



I love to *sing* her songs! Do you?



Not really. I cannot *sing* well.



How about dancing? I love *dancing*!

[Tang et al., ACL'19]

[Lin et al., pre-print'20]

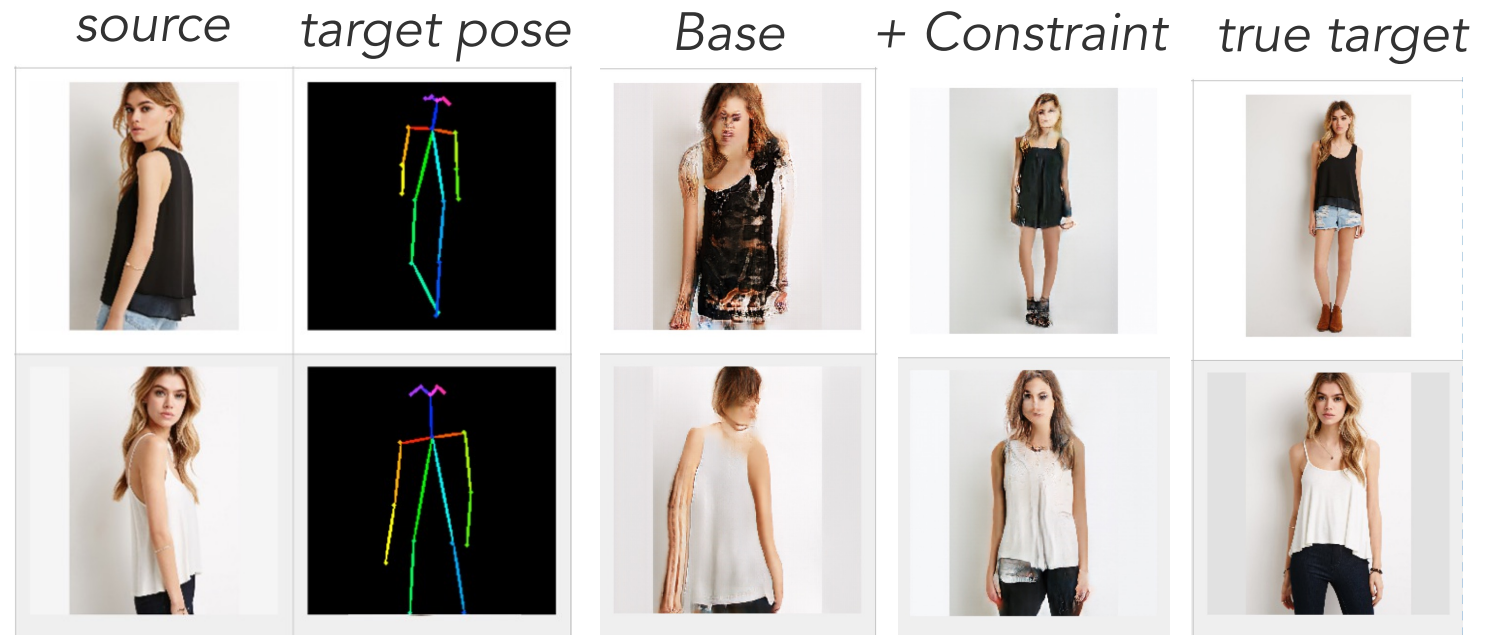
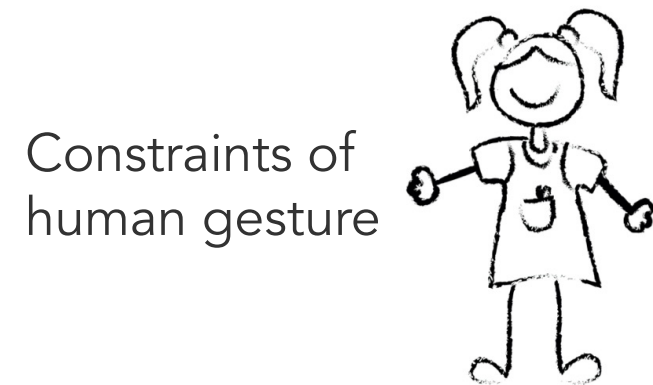


Applications in other controllable generation problems



Source

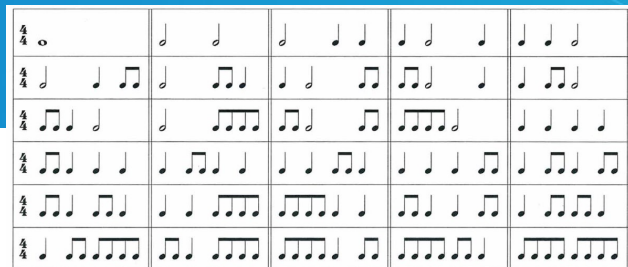
Generated images under different poses



Compositionality



One-off design and programming



Melodic Minor

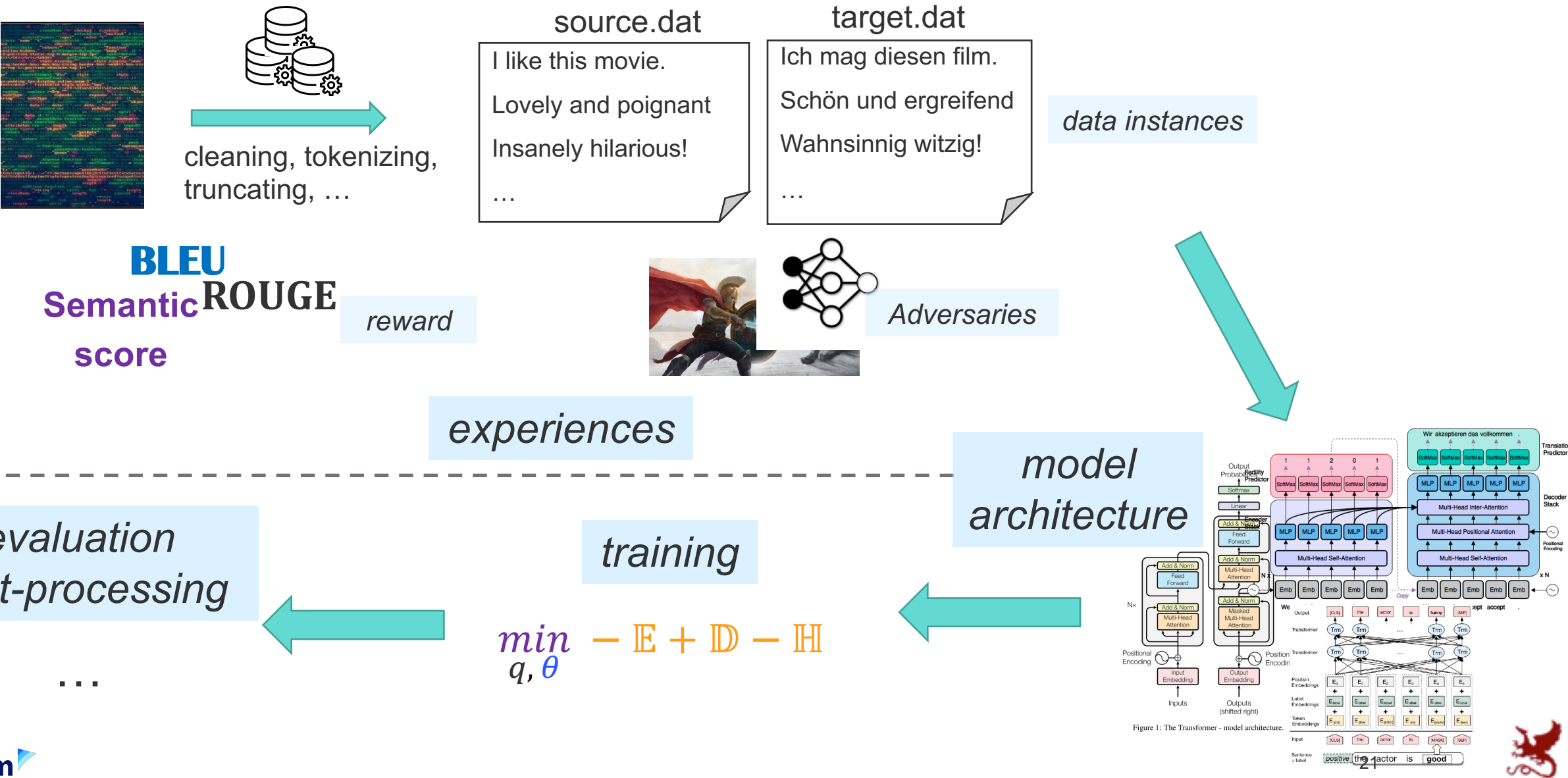
A m	B m	C +	D	E	F #dim	G #dim	A m
I	ii	III +	IV	V	vi.dim	vii.dim	I

A m	G	F	E m	D m	C	B dim	A m
I	VII	VI	v	iv	III	ii.dim	I

Modular and standardized

Complex “rhythms” and “chords”
systematically built out of simpler “notes”

Running Example: Machine Translation (MT)



cleanin
truncat

BLEU
Semantic ROUGE
score

evaluation
post-processing

target.dat

[illegible]

```

235 format_host_command(buf, sizeof buf,
236 {
237     if (sub_command(buf)) {
238         return %s %s\n, sub_error,
239             printf(stderr, "error: %s",
240                 sub_error);
241     }
242 }
243
244 /* Allow a command to be run after
245    while (1) { add host-for-device ...
246    }
247 */
248 if (argc > 3)
249     goto sup;
250
251 return 0;
252 }
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
118
```

[illegible]

```

0 /* no verify APK */};

1
2
3};

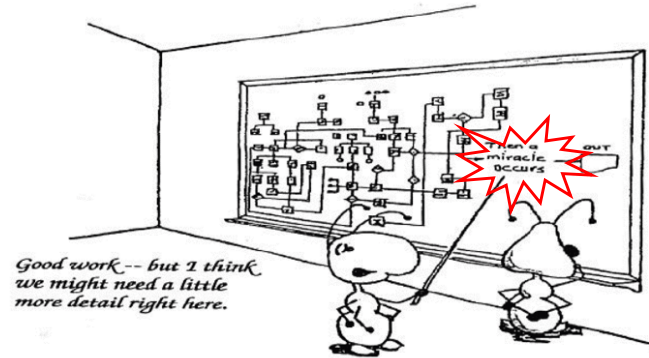
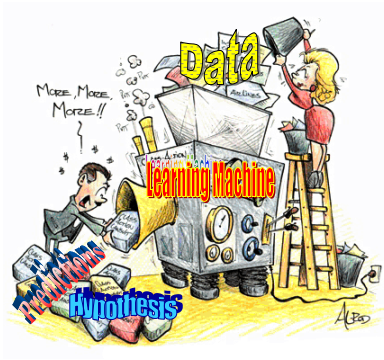
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1
```

The diagram illustrates the Transformer model architecture. It starts with an input sequence (represented by colored circles) which is processed by an embedding layer to produce 'Output Embedding (shifted right)'. This is followed by a series of 'N' identical layers. Each layer contains a 'Masked Multi-Head Attention' block, followed by a residual connection and a layer normalization ('Add & Norm'). The output then goes through a 'Multi-Head Attention' block, another residual connection and layer normalization ('Add & Norm'), and finally a 'Feed Forward' block. The output of the feed-forward block is added to the input of the layer via a residual connection and passed through another layer normalization ('Add & Norm'). The final output is processed by a 'Softmax' layer to produce the 'Output Probability' and a 'Regularity Predictor'.

Diagram illustrating a Transformer-based sentiment classification architecture. The input sentence "positive the actor is" is tokenized into "[CLS]", "[pos]", "[the]", "[actor]", "[is]", and "[CLS]". Each token is passed through an embedding layer (E_pos, E_the, E_actor, E_is) and added to a position embedding (E_1, E_2, E_3, E_4). The resulting sum is passed through a token embedding layer (Trm). The output of the token embedding layer is fed into a Multi-Head Self-Attention layer. The output of the self-attention layer is passed through five MLP layers. The output of the MLP layers is passed through five SoftMax layers. The output of the SoftMax layers is passed through a final SoftMax layer. The final output is "positive".

The diagram illustrates a sequence-to-sequence model architecture for machine translation. The input sequence "Wir akzeptieren das vollkommen" is processed by a stack of N identical layers. Each layer contains three main components: Multi-Head Self-Attention, Multi-Head Positional Attention, and Multi-Head Inter-Attention. The output of the stack is fed into a Translation Predictor, which consists of five SoftMax layers. The final output is "good".

Solution with Composable ML Tools



Composable ML



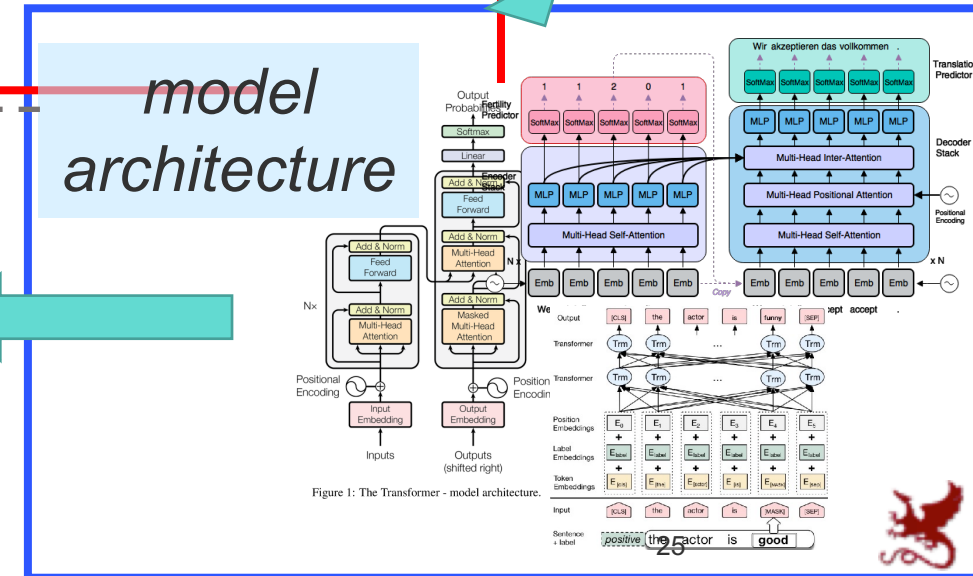
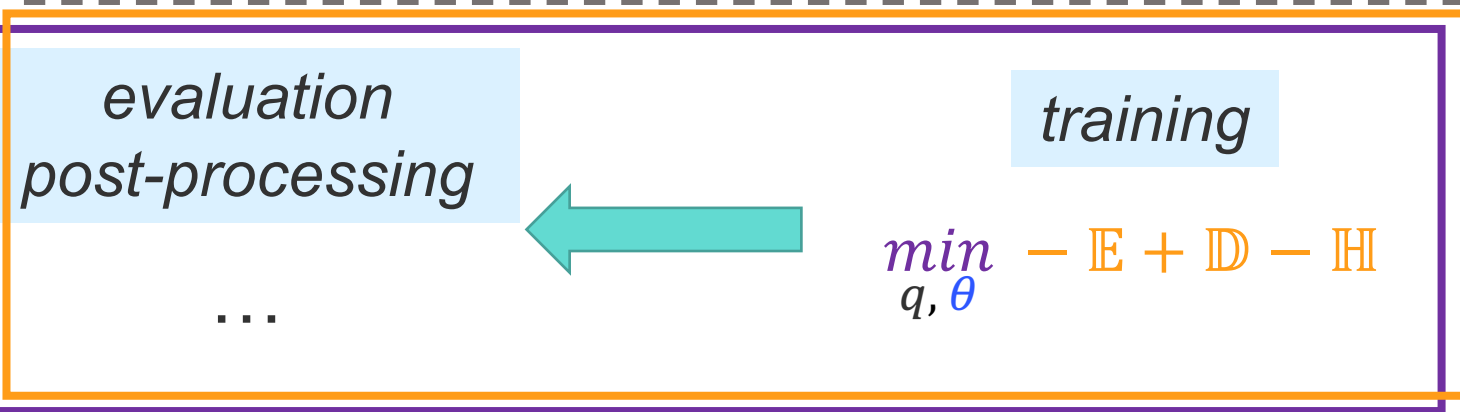
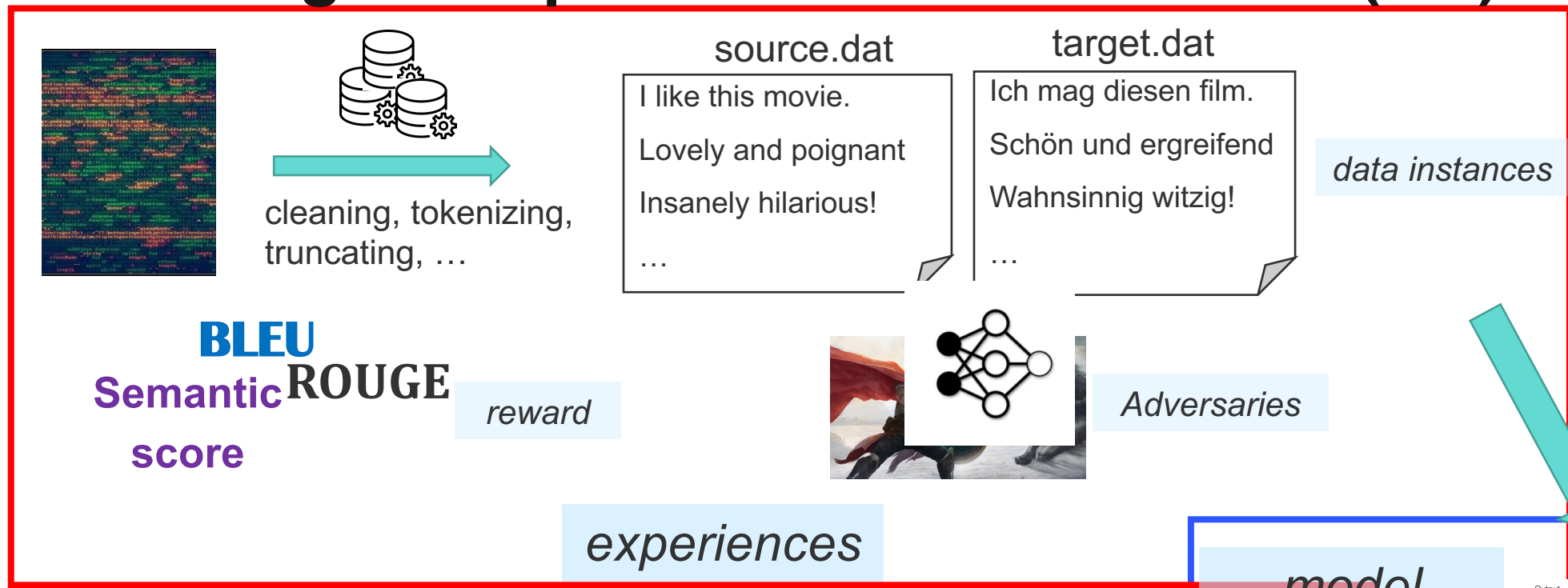
A **modularized** way to build complex applications

Solution with Composable ML Tools

- Build AI solutions more easily, via pick-and-choose:
 - Experiences: data instances, reward, adversaries, rules, ...
 - Models: recurrent, transformer, convolutional, ...
 - Tasks: classification, regression, generation, discovery, ...
- Stop writing same one-off code again and again
 - More reliable and easier to debug
 - Easier to onboard new developers



Running Example: Machine Translation (MT)



ML Components

Optimization

Loss

Divergence

Experience

Operation

Architecture



ML Components

Optimization

Loss

Divergence

Experience

Operation

Architecture

$$\min_{\theta} \mathcal{L}(\theta, \mathcal{E})$$





Architecture

- Neural network design
- Graphical model design
- Compositional architectures





Architecture

- Neural network design
- Graphical model design
- Compositional architectures





Neural Architecture (1): Language Model

- Calculates the probability of a sentence:
 - Sentence:

$$\mathbf{y} = (y_1, y_2, \dots, y_T)$$

Example:

(I, like, this, ...)





Neural Architecture (1): Language Model

- Calculates the probability of a sentence:
 - Sentence:

$$\mathbf{y} = (y_1, y_2, \dots, y_T)$$

$$p_{\theta}(\mathbf{y}) = \prod_{t=1}^T p_{\theta}(y_t \mid \mathbf{y}_{1:t-1})$$

Example:

(I, like, this, ...)

$\dots p_{\theta}(\text{like} \mid I) p_{\theta}(\text{this} \mid I, \text{like}) \dots$





Neural Architecture (1): Language Model

- Calculates the probability of a sentence:
 - Sentence:

$$\mathbf{y} = (y_1, y_2, \dots, y_T)$$

$$p_{\theta}(\mathbf{y}) = \prod_{t=1}^T p_{\theta}(y_t | \mathbf{y}_{1:t-1})$$

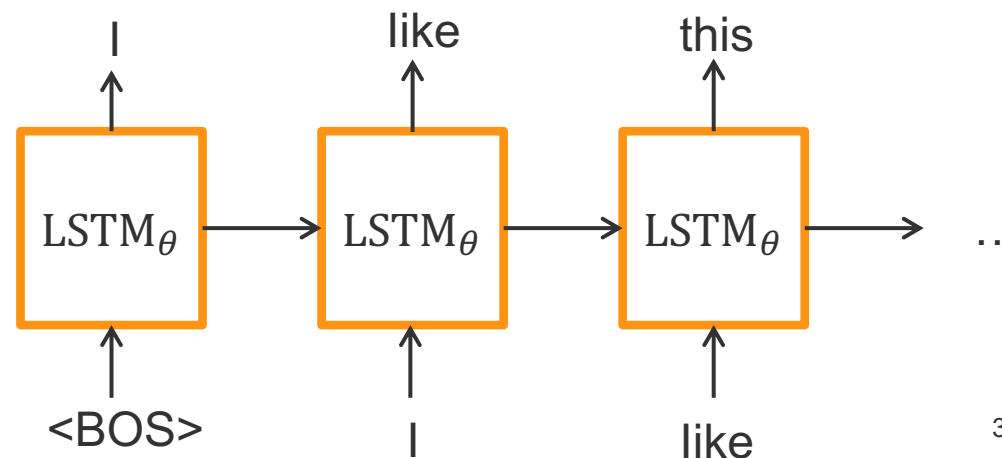
Example:

(I, like, this, ...)

$\dots p_{\theta}(\text{like} | I) p_{\theta}(\text{this} | I, \text{like}) \dots$

Architecture (1.1)

LSTM RNN





Neural Architecture (1): Language Model

- Calculates the probability of a sentence:
 - Sentence:

$$\mathbf{y} = (y_1, y_2, \dots, y_T)$$

$$p_{\theta}(\mathbf{y}) = \prod_{t=1}^T p_{\theta}(y_t \mid \mathbf{y}_{1:t-1})$$

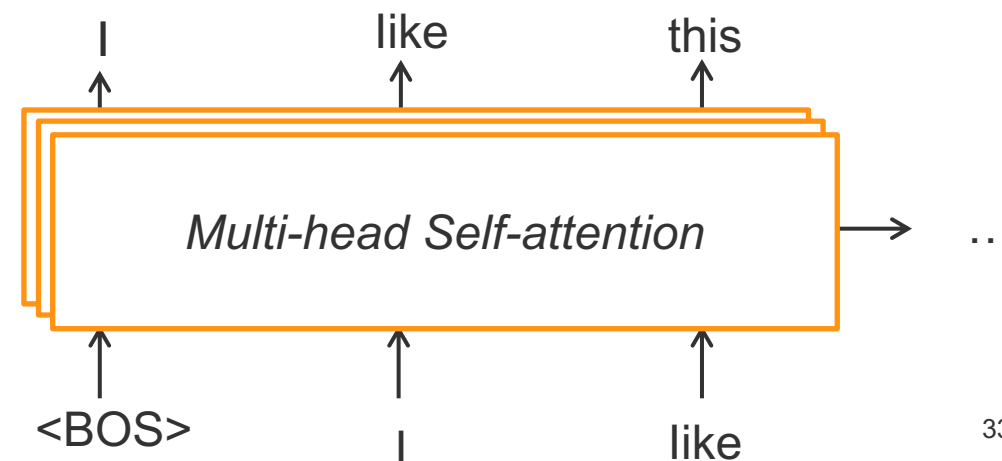
Example:

(I, like, this, ...)

$\dots p_{\theta}(\text{like} \mid I) p_{\theta}(\text{this} \mid I, \text{like}) \dots$

Architecture (1.2)

Transformer

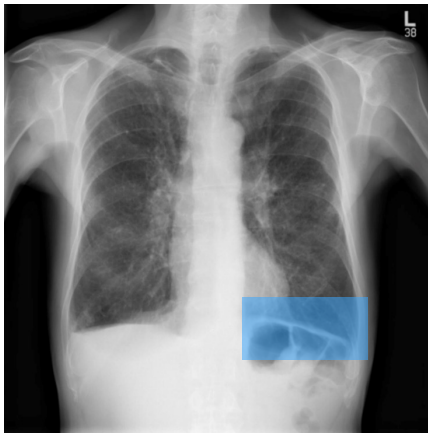


Neural Architecture (2): **Conditional** Language Model

- Conditions on additional task-dependent context x
 - Machine translation: source sentence

I like this movie. → Ich mag diesen film.

- Medical image report generation: medical image



... There is chronic pleural-
parenchymal scarring within
the lung bases. No lobar
consolidation is seen. ...

Neural Architecture (2): **Conditional** Language Model

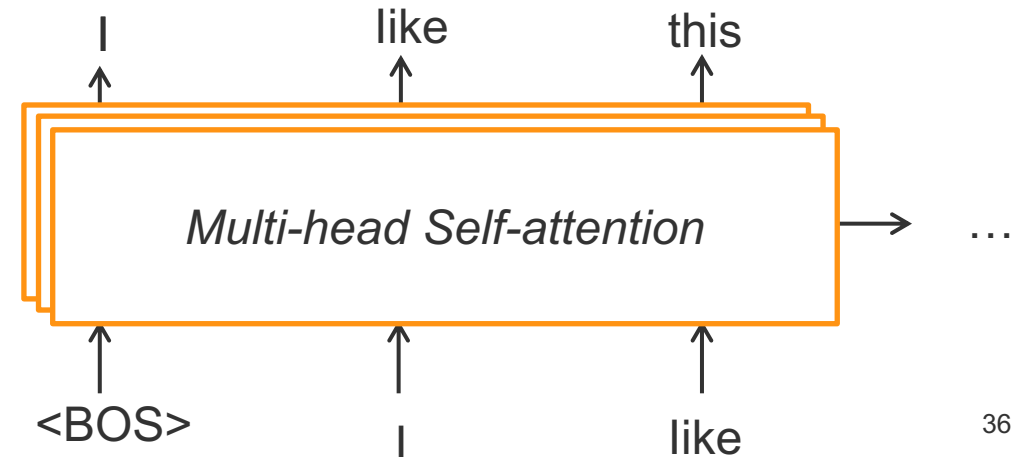
- Conditions on additional task-dependent context $\mathbf{x}$

$$p_{\theta}(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^T p_{\theta}(y_t \mid \mathbf{y}_{1:t-1}, \mathbf{x})$$

Neural Architecture (2): **Conditional** Language Model

- Conditions on additional task-dependent context $\mathbf{x}$

$$p_{\theta}(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^T p_{\theta}(y_t \mid \mathbf{y}_{1:t-1}, \mathbf{x})$$

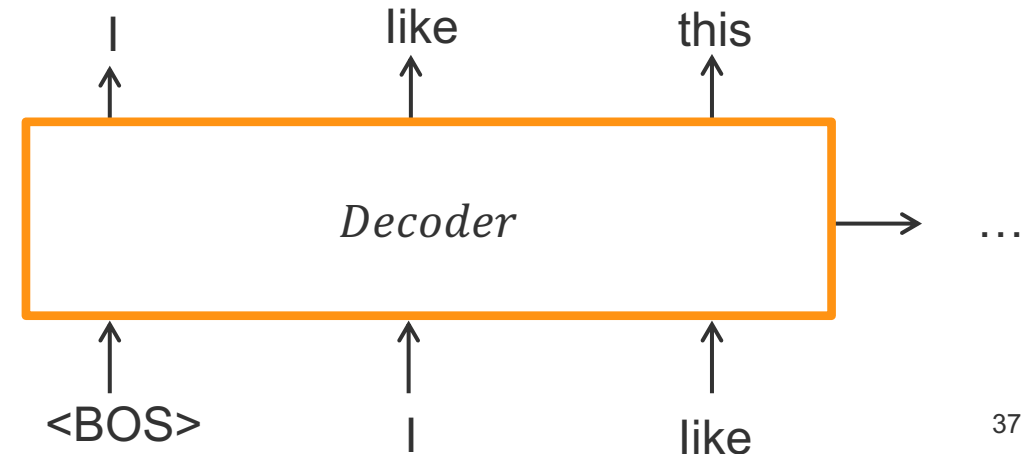


Neural Architecture (2): **Conditional** Language Model

- Conditions on additional task-dependent context $\mathbf{x}$

$$p_{\theta}(\mathbf{y} | \mathbf{x}) = \prod_{t=1}^T p_{\theta}(y_t | \mathbf{y}_{1:t-1}, \mathbf{x})$$

- Language model as a **decoder**

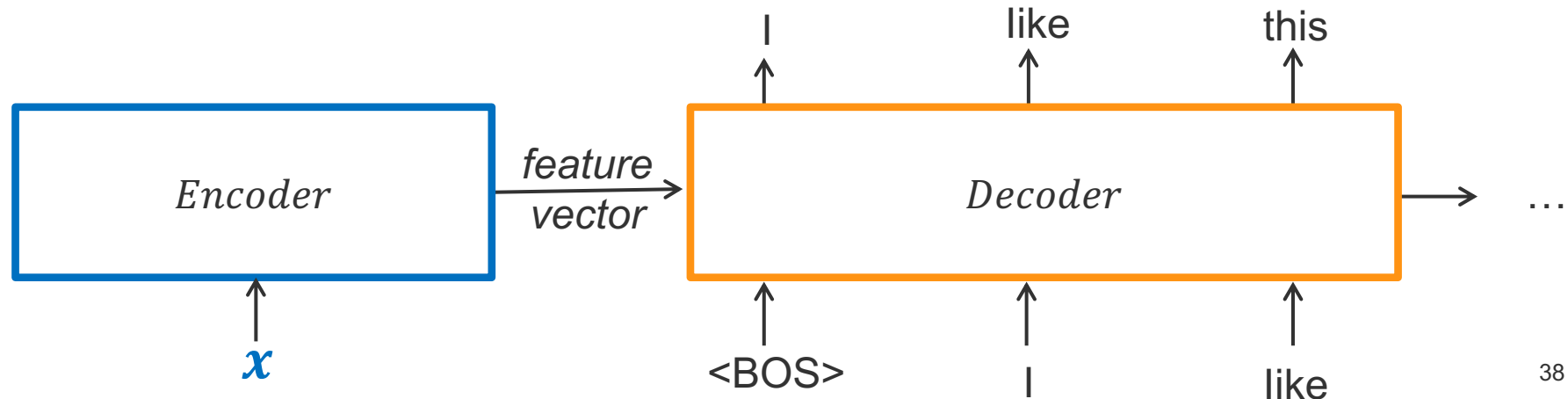


Neural Architecture (2): **Conditional** Language Model

- Conditions on additional task-dependent context $\mathbf{x}$

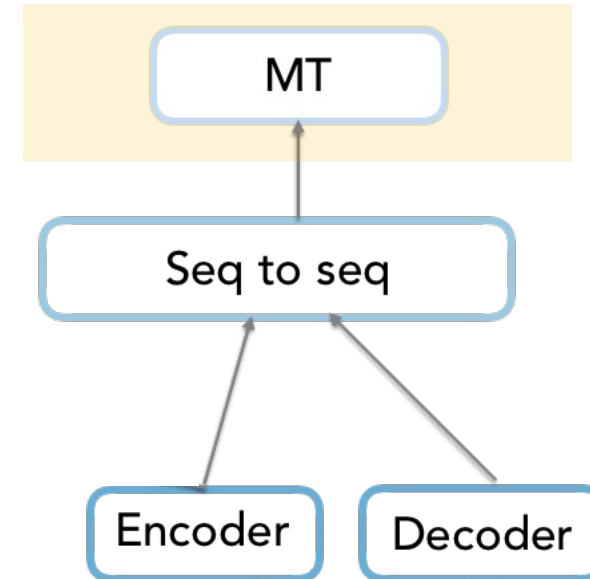
$$p_{\theta}(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^T p_{\theta}(y_t \mid \mathbf{y}_{1:t-1}, \mathbf{x})$$

- Language model as a **decoder**
- Encodes context with an **encoder**



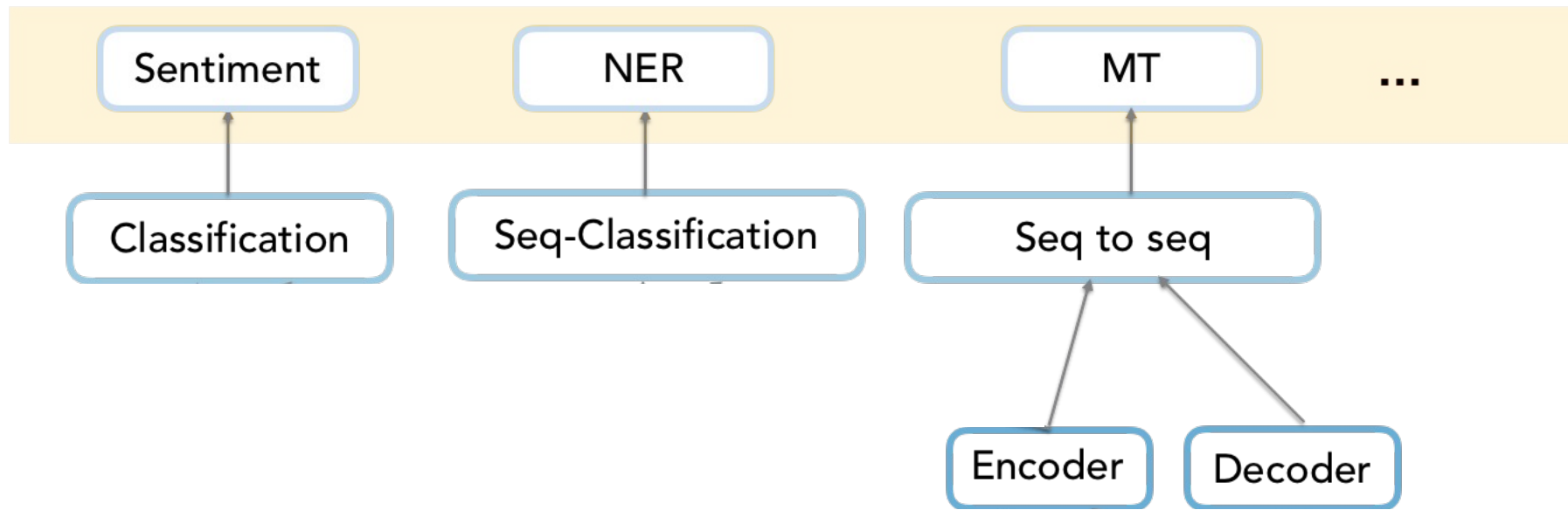
Neural Architecture Components

Task:



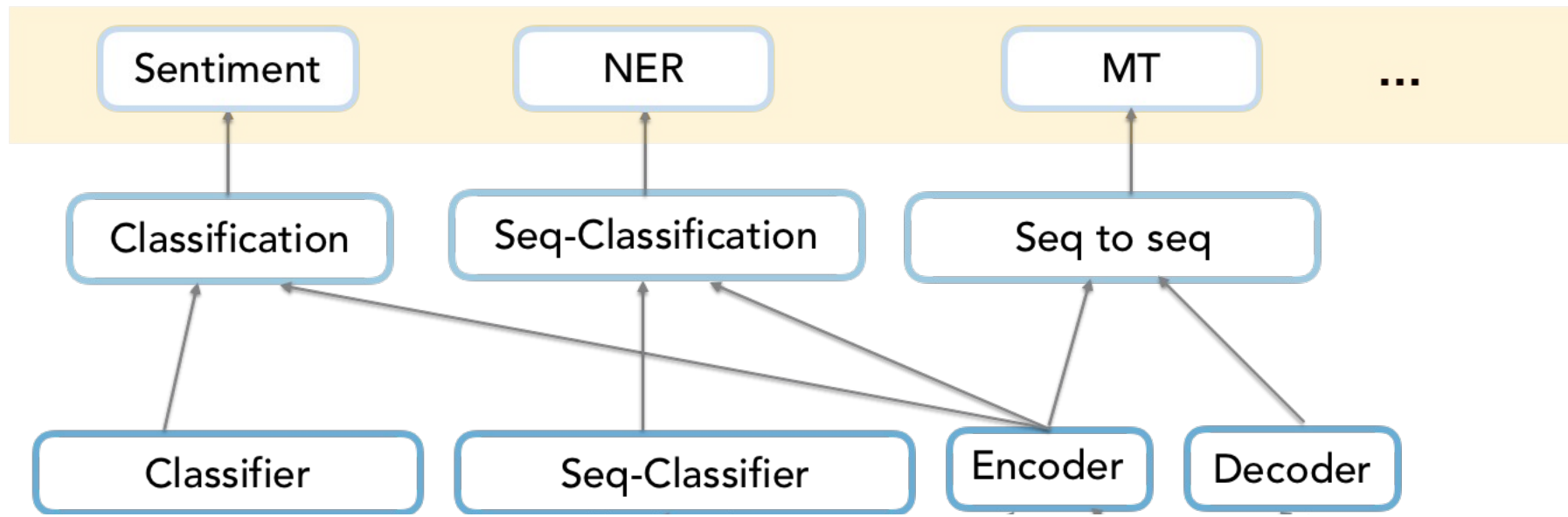
Neural Architecture Components

Task:



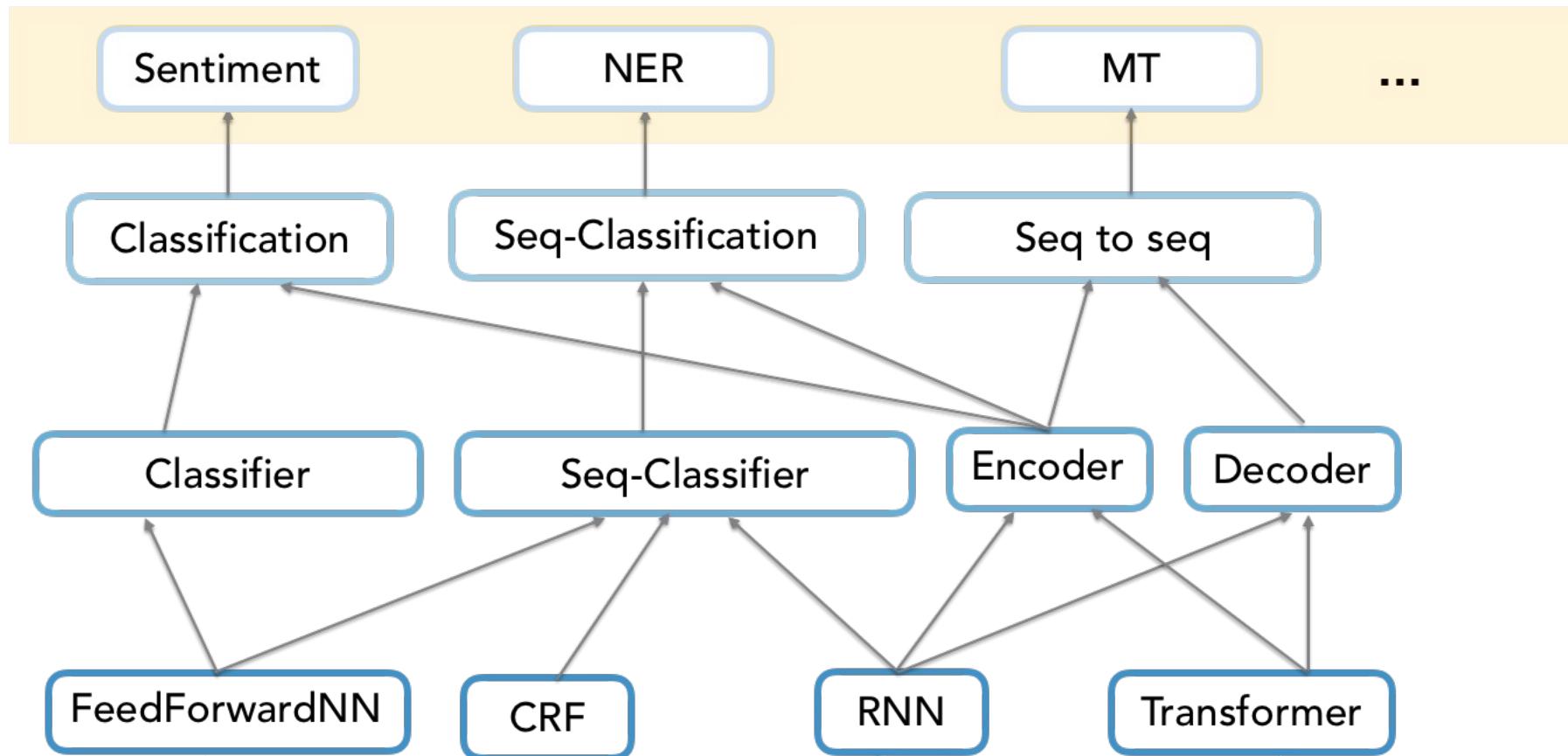
Neural Architecture Components

Task:



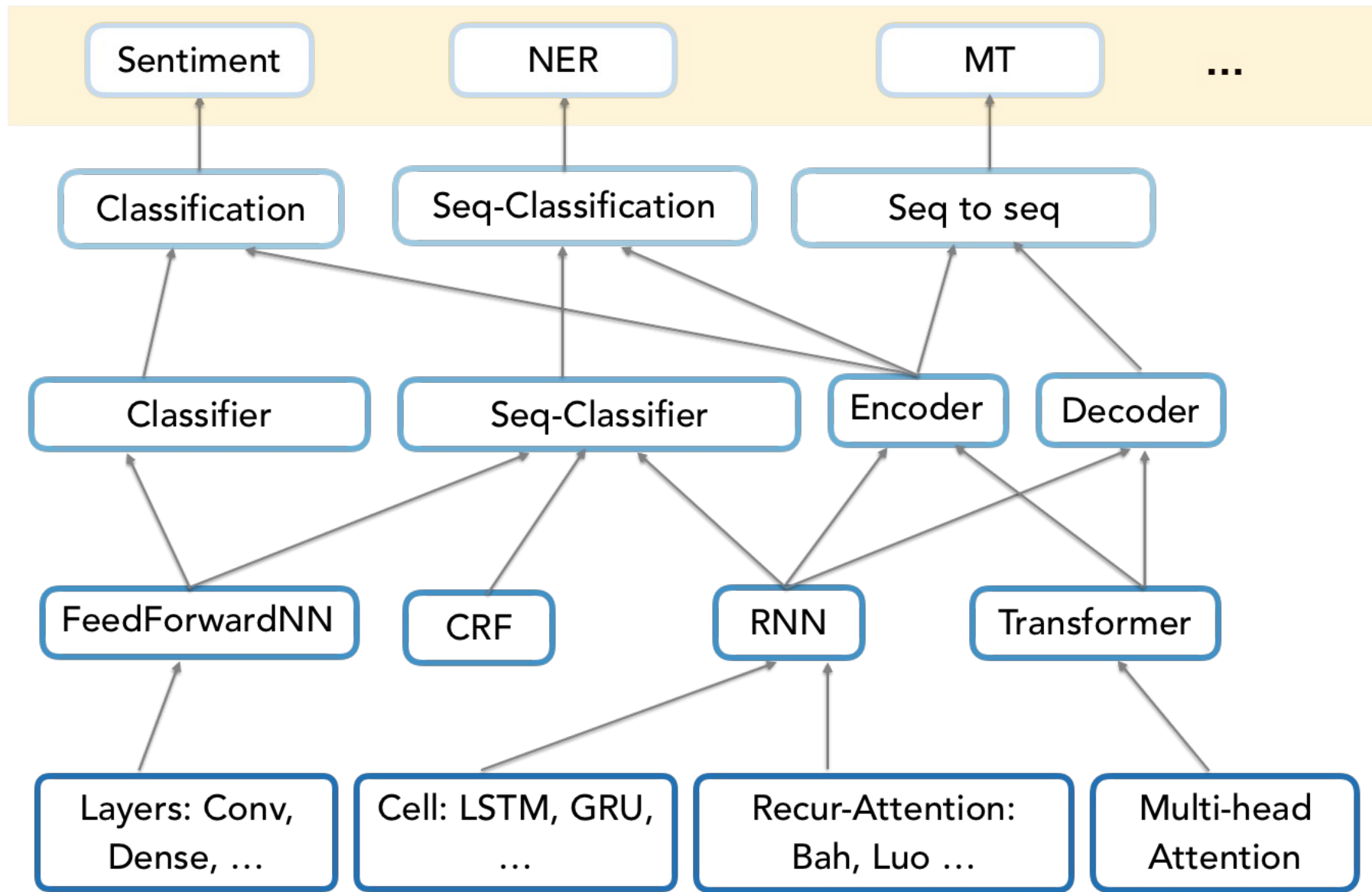
Neural Architecture Components

Task:



Neural Architecture Components

Task:

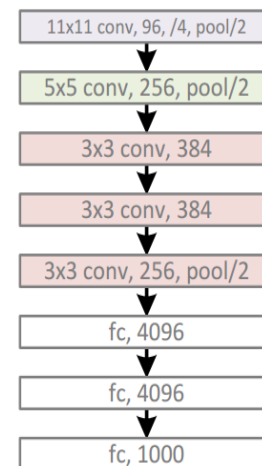




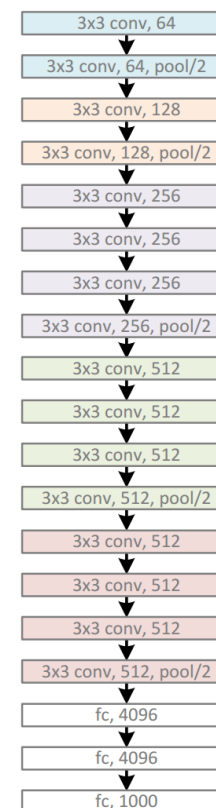
Other Neural Architecture Components

- Layers
 - Fully connected
 - Convolutional & pooling
 - Recurrent
 - ResNets
 - Etc.
- Activation functions
 - Linear and ReLU
 - Sigmoid and tanh
 - Etc.

AlexNet
8 layers



VGG
19 layers



GoogleNet
22 layers



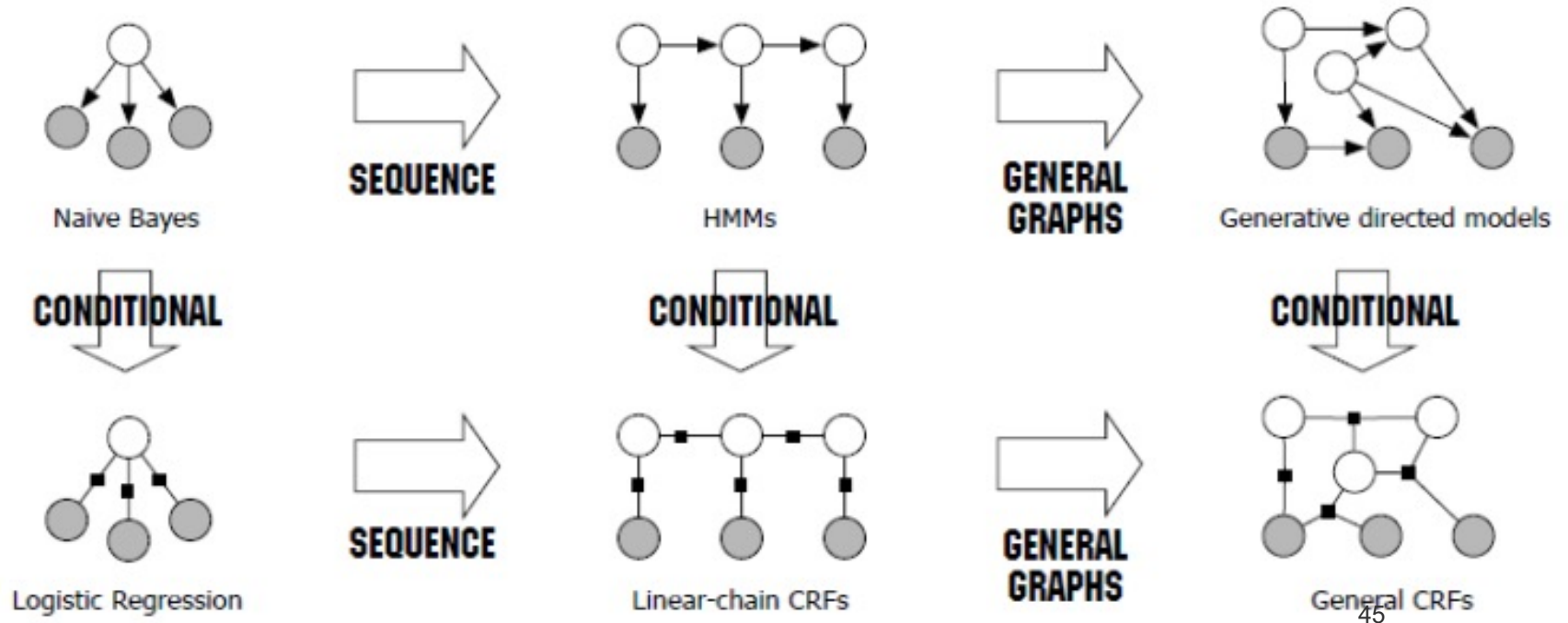
ResNet
152 layers





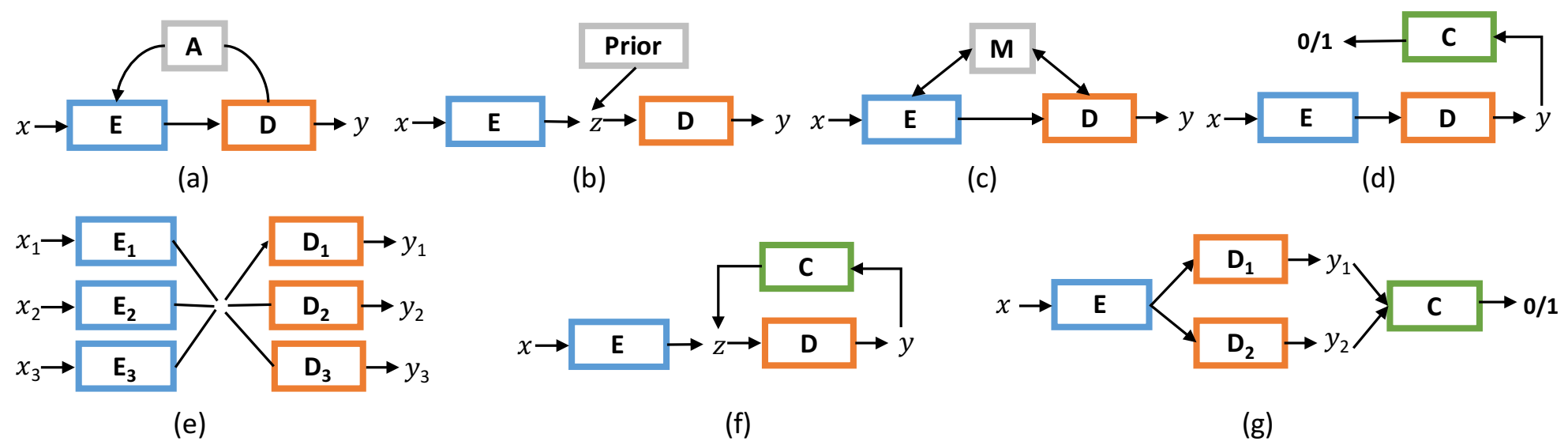
Architecture

- Neural network design
- Graphical model design
- Compositional



Architecture

- Neural network design
- Graphical model design
- Compositional architectures





ML Components

Optimization

Loss

Divergence

Experience

Operation

Architecture

decoder

LSTM RNN

Attention RNN

Transformer

...

encoder

classifier





ML Components

Optimization

Loss

Divergence

Experience

Operation

Architecture

decoder

LSTM RNN

Attention RNN

Transformer

...

encoder

classifier





Operation, Loss & Optimization (1): Learning from data instances

Loss

- Experience: data instances $(\mathbf{x}^*, \mathbf{y}^*)$
- Divergence: cross entropy
- Reduce to MLE:

$$\mathcal{L}_{\text{MLE}} = -\log p_{\theta}(\mathbf{y}^* | \mathbf{x}^*) = -\prod_{t=1}^T p_{\theta}(y_t^* | \mathbf{y}_{1:t-1}^*, \mathbf{x}^*)$$

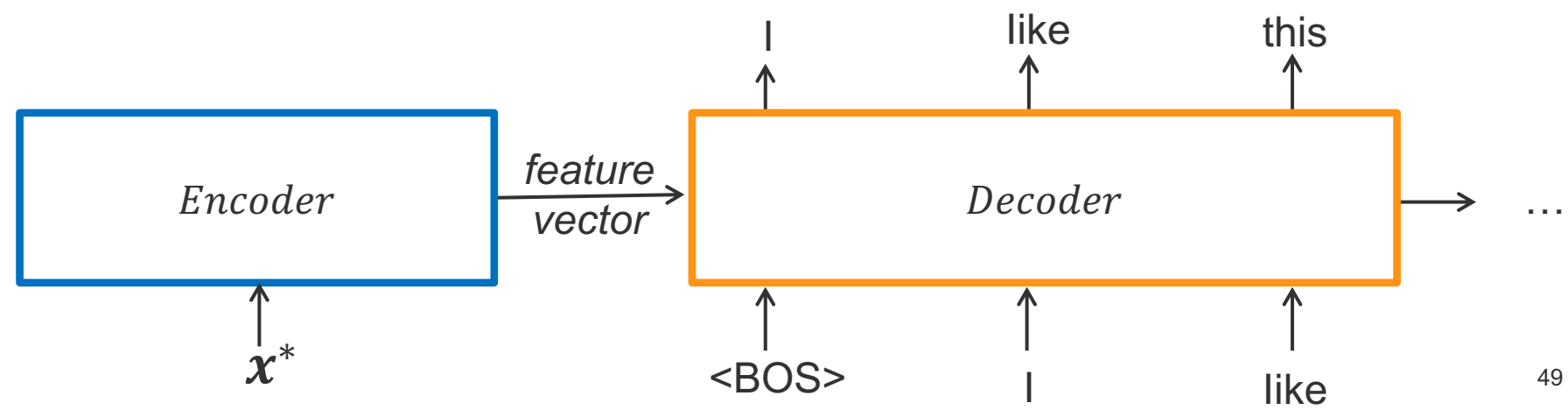
Optimization

SGD

Operation

Teacher-forcing decoding:

For every step t , feeds in the previous ground-truth tokens $\mathbf{y}_{1:t-1}^*$ to decode next step





Operation, Loss & Optimization (2): Learning from adversaries + data instances

Loss

- Experience:
 - data instances $(\mathbf{x}^*, \mathbf{y}^*)$
 - adversary (discriminator)
- Divergence: JSD, f-Div., W-dist.

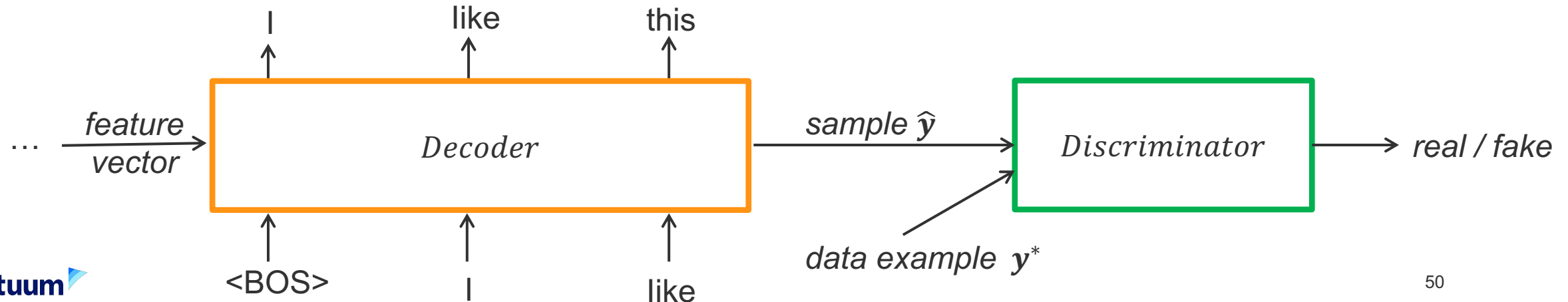
Operation

Gumbel-softmax decoding:

Uses a differentiable approximation of sample $\hat{\mathbf{y}}$ for gradient backpropagation

$$\frac{\partial \mathcal{L}(\hat{\mathbf{y}})}{\partial \theta} = \frac{\partial \mathcal{L}(\hat{\mathbf{y}})}{\partial \hat{\mathbf{y}}} \frac{\partial \hat{\mathbf{y}}}{\partial \theta}$$

Optimization PFD, convex duality





Operation, Loss & Optimization (3): Learning from reward + data instances

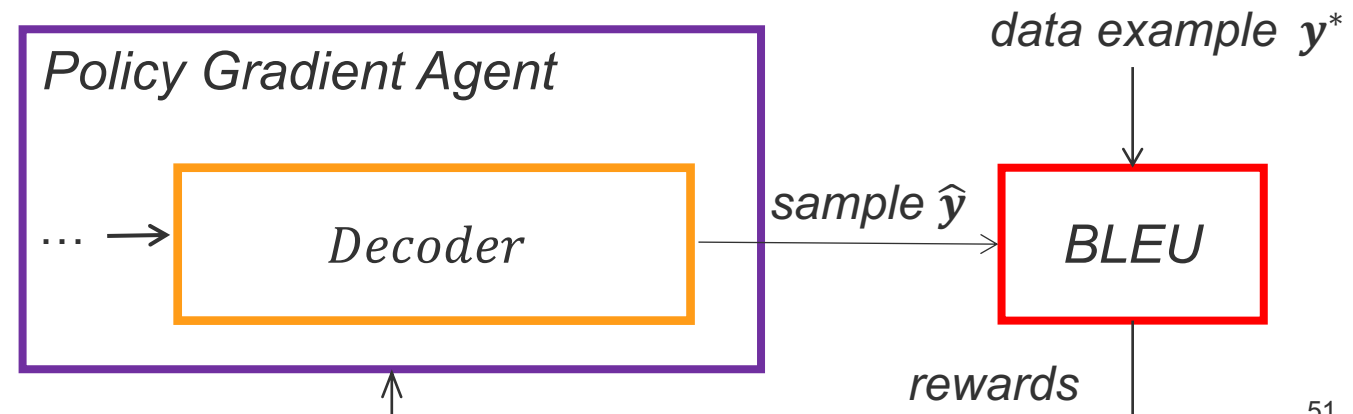
Loss

- Experience:
 - data instances (x^*, y^*)
 - reward metric, e.g., BLEU
- Divergence: cross entropy

Operation

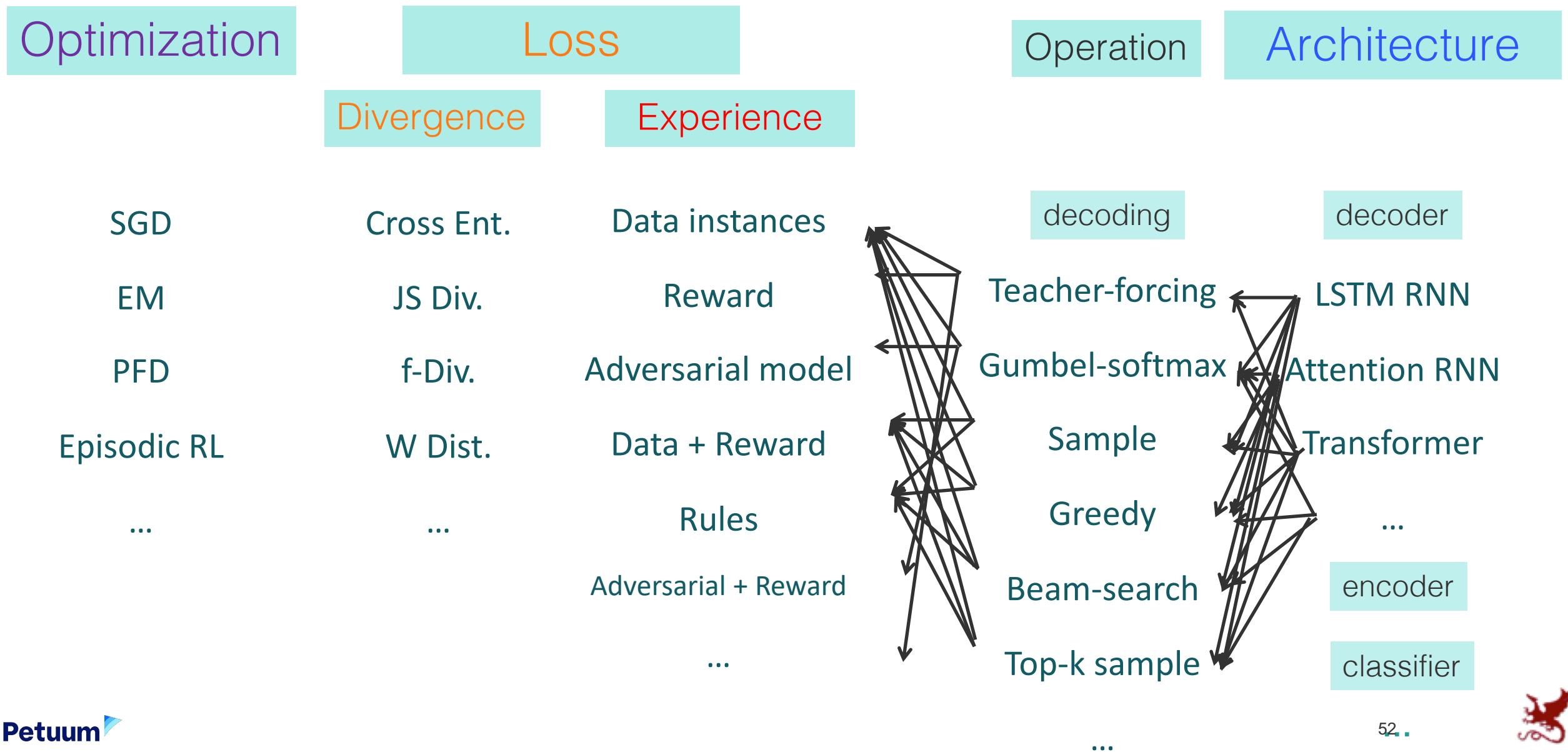
- Greedy decoding
- Sampling decoding
- Beam search decoding
- Top- k / Top- p decoding
- ...

Optimization EM





Holistic View

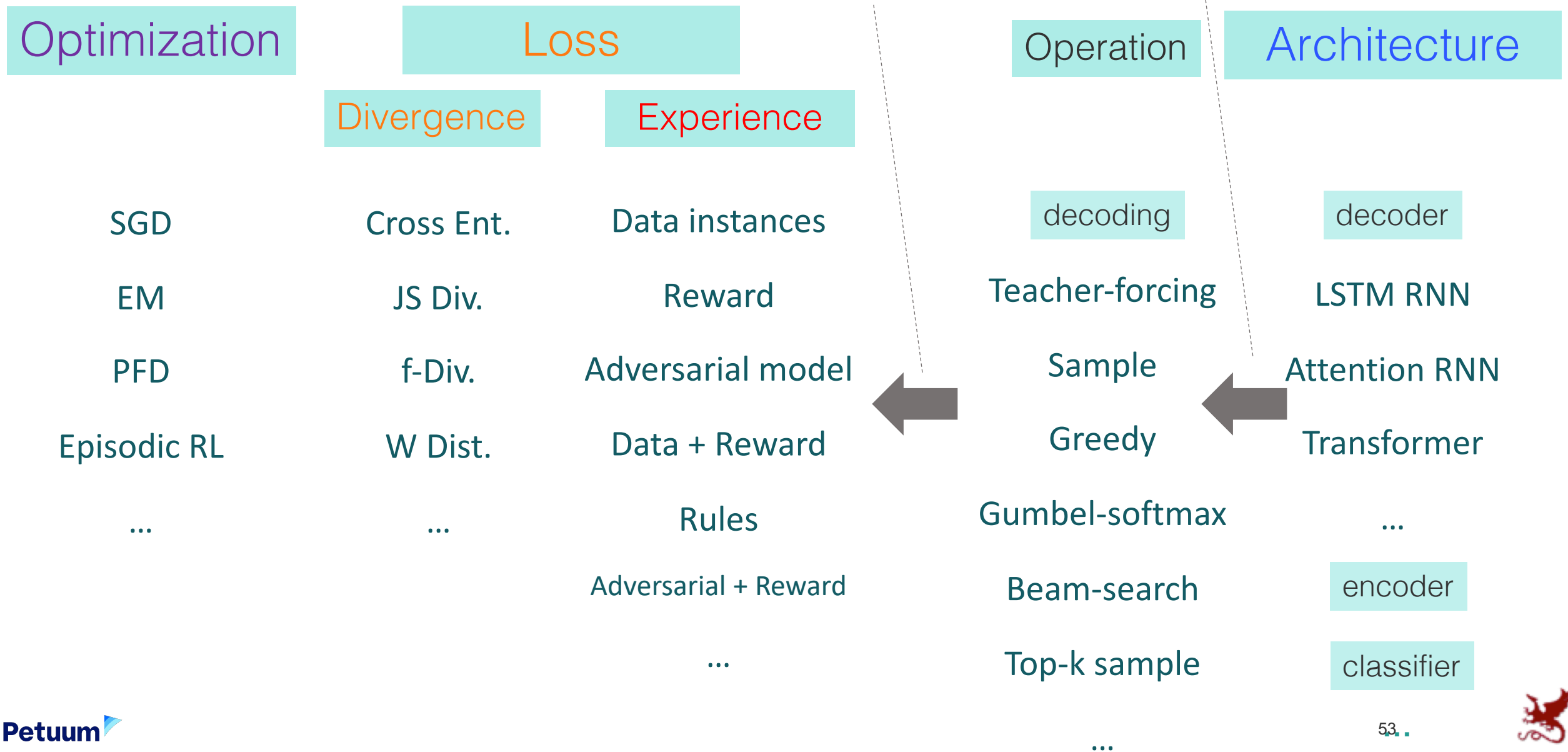




Holistic View

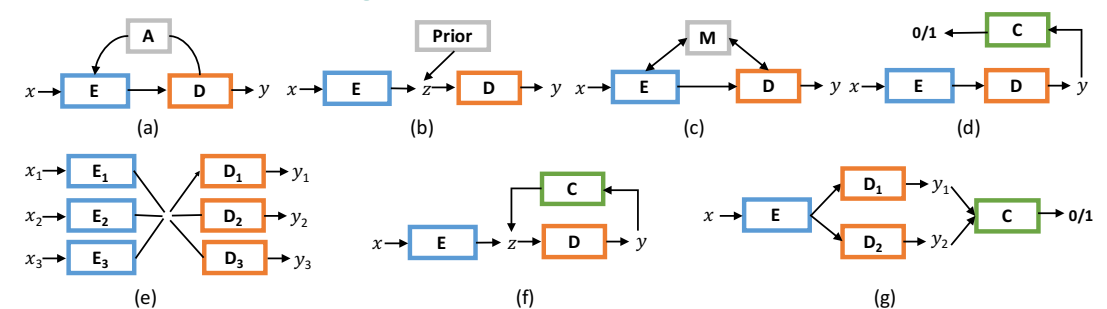
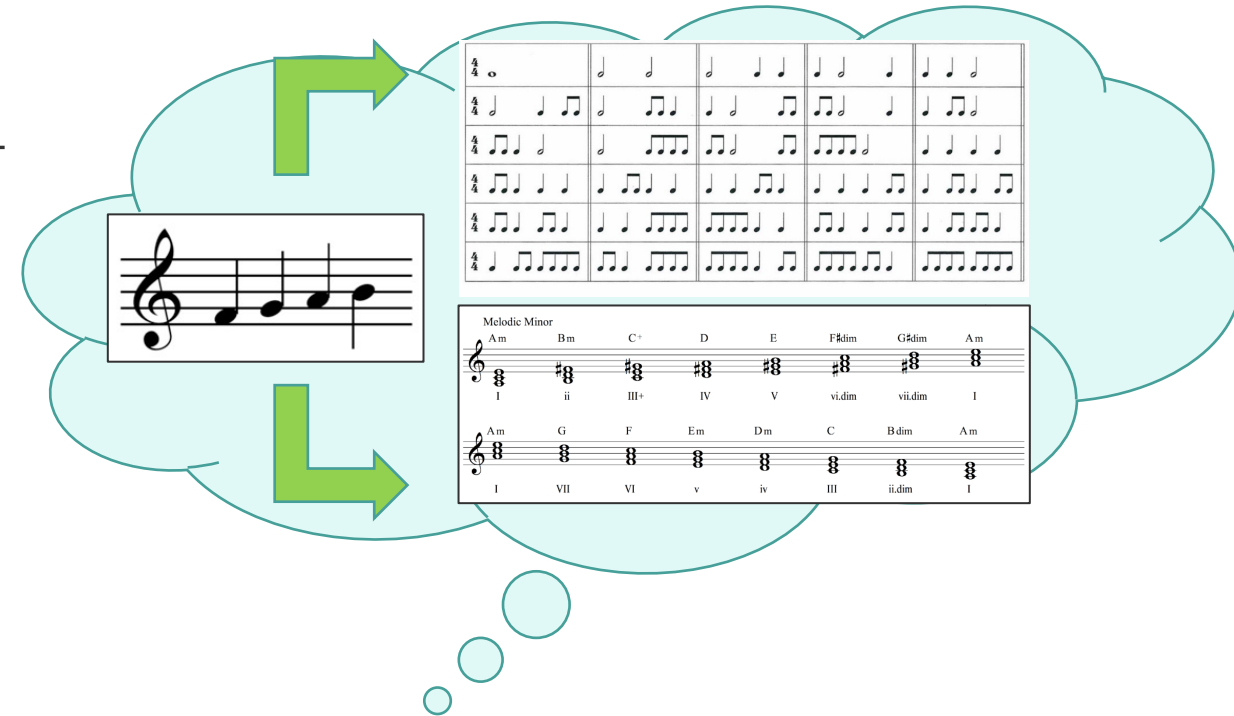
*called as
subroutines*

*uniform
interfaces*

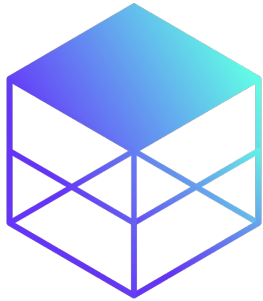


Summary so far: the concept of composable ML

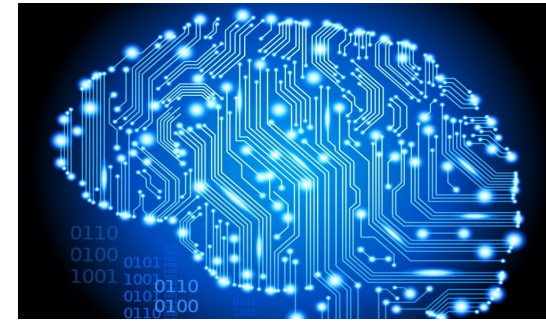
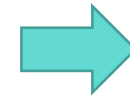
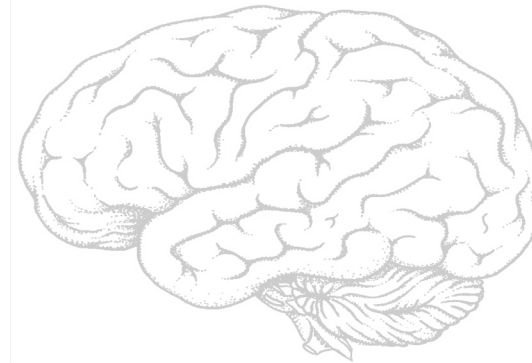
- Composable ML
 - Basic “musical notes” for complex ML systems
 - Structure/rules for combining notes into “chords”/“rhythms” ...
 - ... and chords/rhythms into compositions
 - (or just think of it as Lego for ML)
- Next – Symbolic programming via Petuum Texar open source



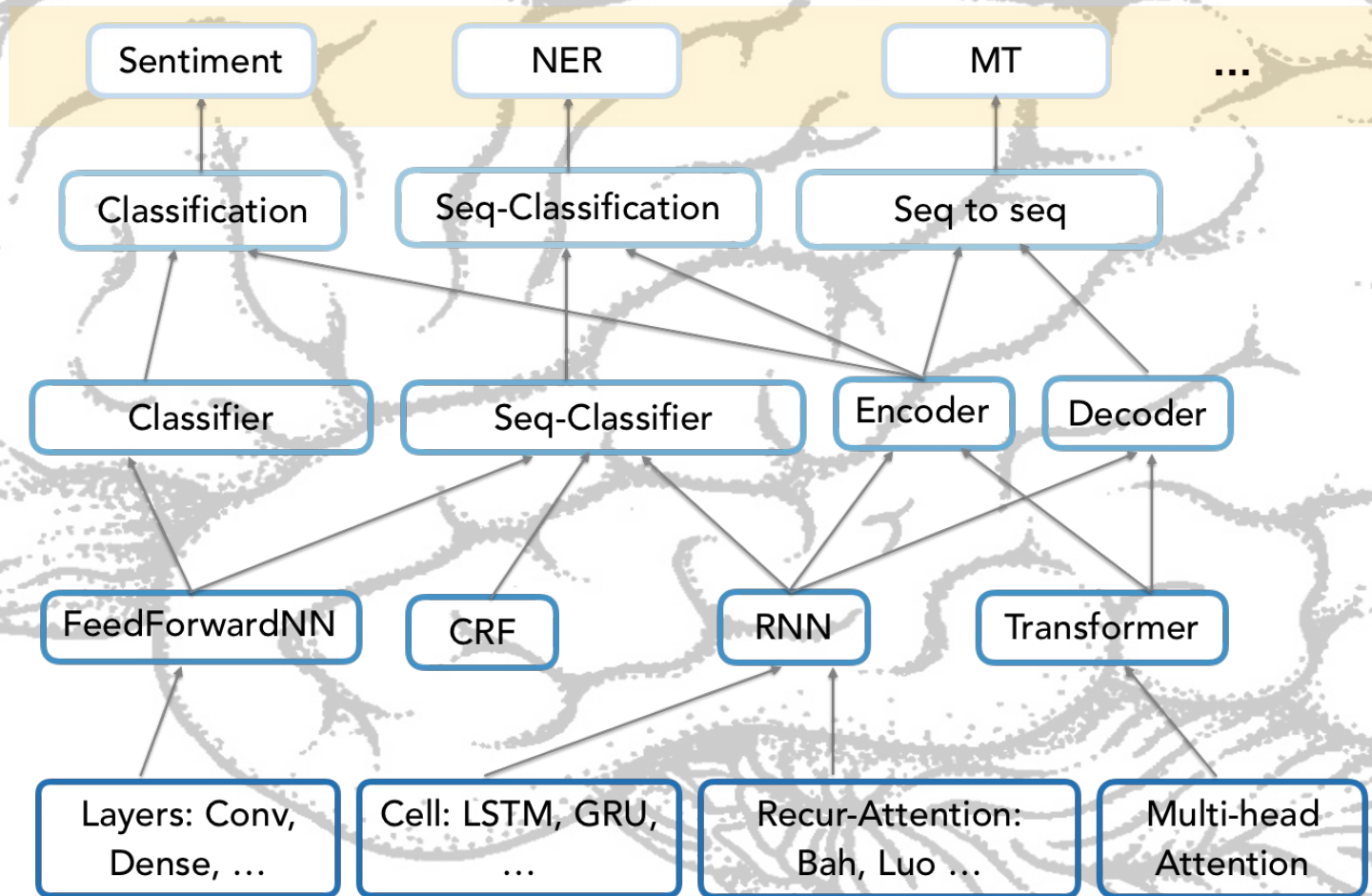
ML Composition with Texar



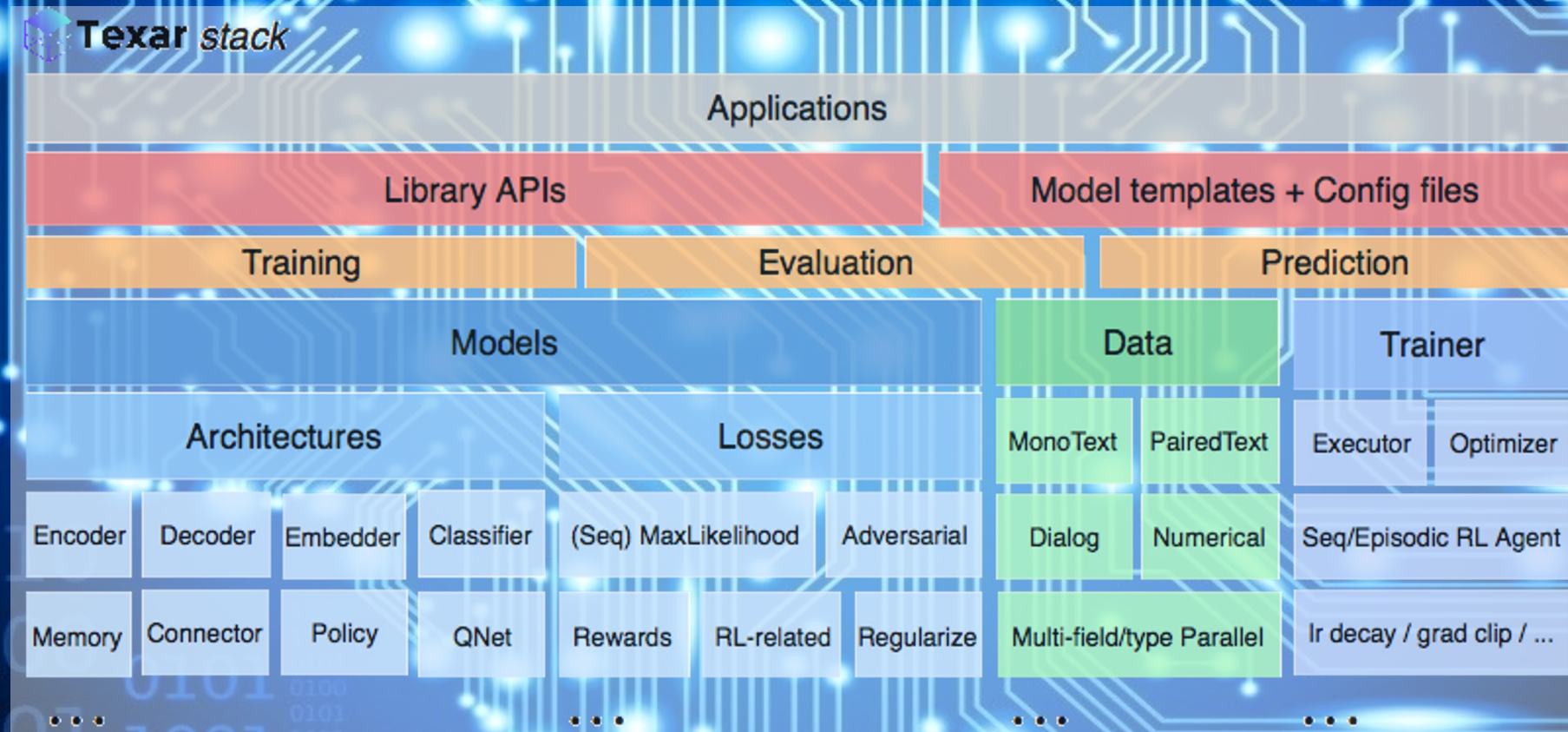
Compose your ML applications
like playing building blocks



Expert's Intellectual “View” of Composable ML

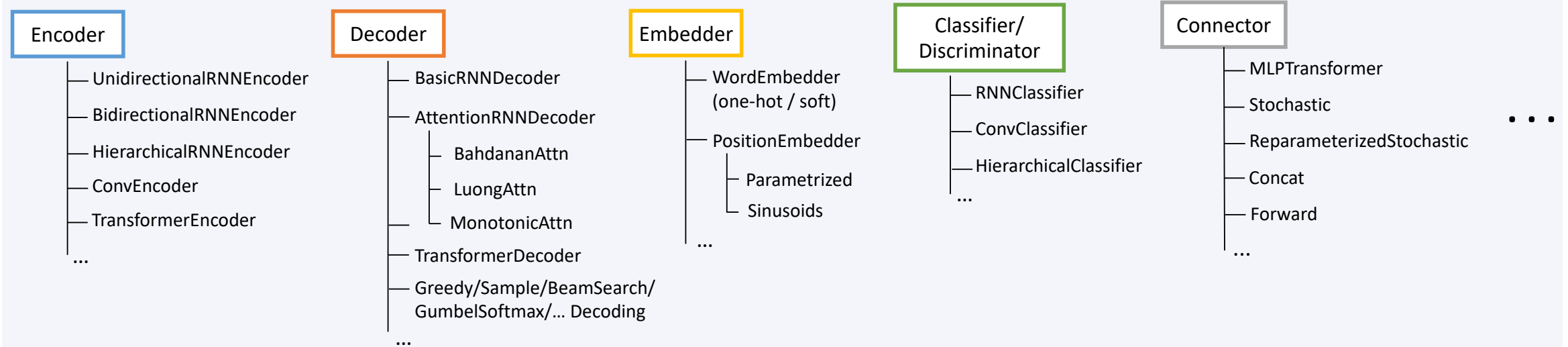


Texar Stack – Operationalized “View” of Composable ML

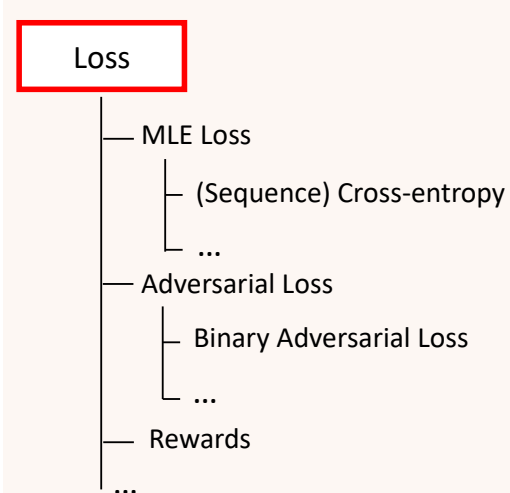


Module Catalog in Texar

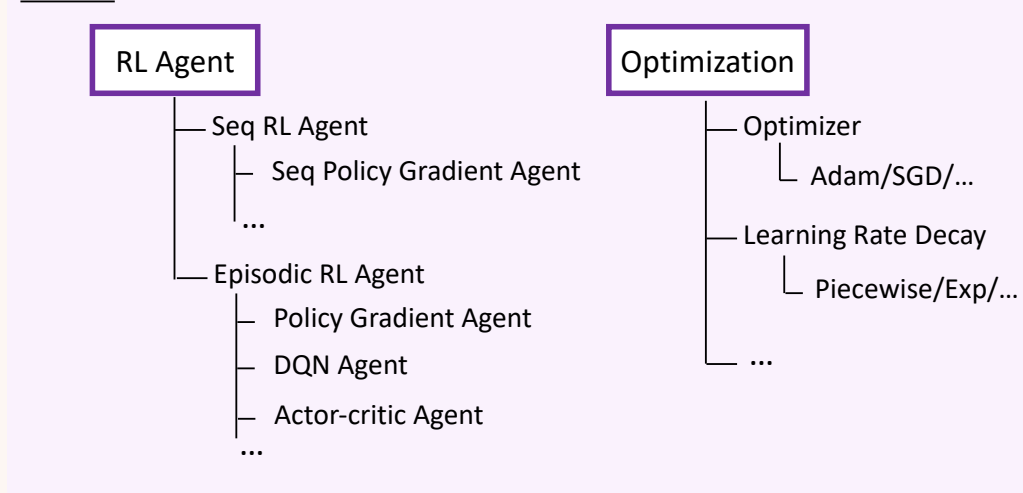
Model architecture



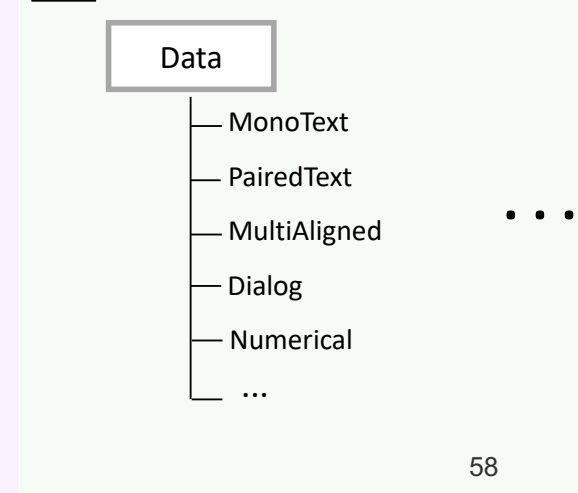
Model loss



Trainer



Data



Texar Highlights



Modularized

Assembles any complex model like playing building blocks



Versatile

Supports a large variety of models/algorithms/applications ...



Extensible

Allows to plug in any customized or external modules

Running Example: Machine Translation (MT)



cleaning, tokenizing,
truncating, ...

source.dat

I like this movie.
Lovely and poignant
Insanely hilarious!
...

target.dat

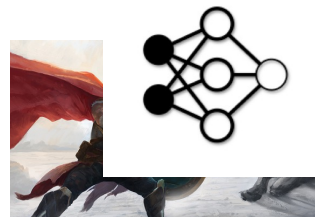
Ich mag diesen film.
Schön und ergreifend
Wahnsinnig witzig!
...

data instances

BLEU
Semantic
score

ROUGE

reward



Adversaries

experiences

*model
architecture*

*evaluation
post-processing*

training

$$\min_{q, \theta} - \mathbb{E} + \mathbb{D} - \mathbb{H}$$

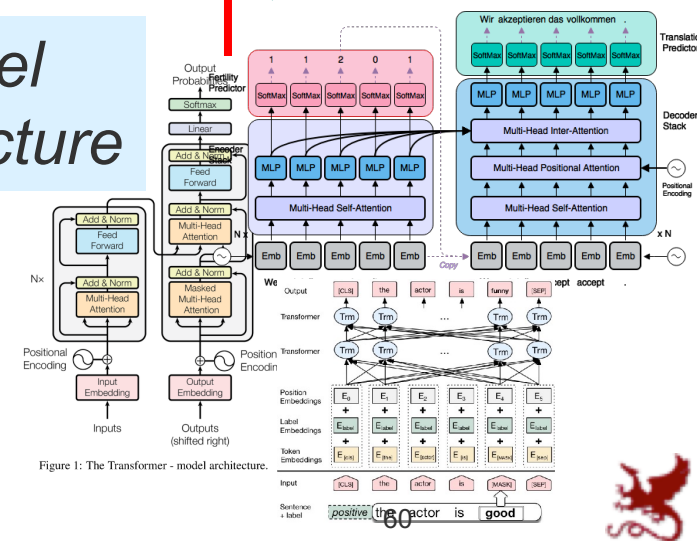


Figure 1: The Transformer - model architecture.

Running Example: MT - Learning from data instances

- Input: a source sentence:
I like this movie.
- Output: a target sentence:
Ich mag diesen film.
- Dataset:

source.txt

I like this movie.
Lovely and poignant
Insanely hilarious!
...

target.txt

Ich mag diesen film.
Schön und ergreifend
Wahnsinnig witzig!
...

vocab.txt

I
like
this
movie
Ich
mag



Running Example: MT - Learning from data instances



Running Example: MT - Learning from data instances

Data



```
1 # Read data
2 dataset = PairedTextData(data_hparams)
3 batch = DataIterator(dataset).get_next()
```

YAML config files

```
1. data_hparams:
2.   batch_size: 64
3.   num_epochs: 10
4.   shuffle: True
5.   source_dataset:
6.     files: 'source.txt'
7.     vocab_file: 'vocab.txt'
8.     max_seq_length: 100
9.     bos_token: '<BOS>'
10.    eos_token: '<EOS>'
11.  target_dataset:
12.    ...
13.  ...
```



Running Example: MT - Learning from data instances

Data

```
1 # Read data
2 dataset = PairedTextData(data_hparams)
3 batch = DataIterator(dataset).get_next()
```

Architecture
& Inference

YAML config files

Running Example: MT - Learning from data instances

Data

```
1 # Read data
2 dataset = PairedTextData(data_hparams)
3 batch = DataIterator(dataset).get_next()
4 # Encode
5 embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams)
```

YAML config files

```
1. embedder_hparams:
2.   embedding_dim: 256
3.   dropout_rate: 0.9
4.   regularization: 'L1L2'
5.   ...
```

Architecture
& Inference



Running Example: MT - Learning from data instances

Data

```
1 # Read data
2 dataset = PairedTextData(data_hparams)
3 batch = Dataloader(dataset).get_next()
4 # Encode
5 embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams)
6 encoder = TransformerEncoder(hparams=encoder_hparams)
```

Architecture
& Inference

YAML config files

```
1. encoder_hparams:
2.   num_blocks: 16
3.   num_heads: 8
4.   hidden_dim: 256
5.   output_dim: 128
6.   dropout_rate: 0.8
7.   ...
```



Running Example: MT - Learning from data instances

Data

```

1 # Read data
2 dataset = PairedTextData(data_hparams)
3 batch = Dataloader(dataset).get_next()
4 # Encode
5 embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams)
6 encoder = TransformerEncoder(hparams=encoder_hparams)
7 enc_outputs = encoder(embedder(batch['source_text_ids']),
8                        batch['source_length'])
    
```

Architecture
& Inference



Running Example: MT - Learning from data instances

Data

```

1  # Read data
2  dataset = PairedTextData(data_hparams)
3  batch = Dataloader(dataset).get_next()
4  # Encode
5  embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams)
6  encoder = TransformerEncoder(hparams=encoder_hparams)
7  enc_outputs = encoder(embedder(batch['source_text_ids']),
8                        batch['source_length'])
9  # Build decoder
10 decoder = AttentionRNNDecoder(memory=enc_outputs,
11                               hparams=decoder_hparams)
12 # Maximum Likelihood Estimation
13 ## Teacher-forcing decoding
14 outputs, length, _ = decoder(decoding_strategy='teacher-forcing',
15                              inputs=embedder(batch['target_text_ids']),
16                              seq_length=batch['target_length']-1)

```

Architecture
& Inference

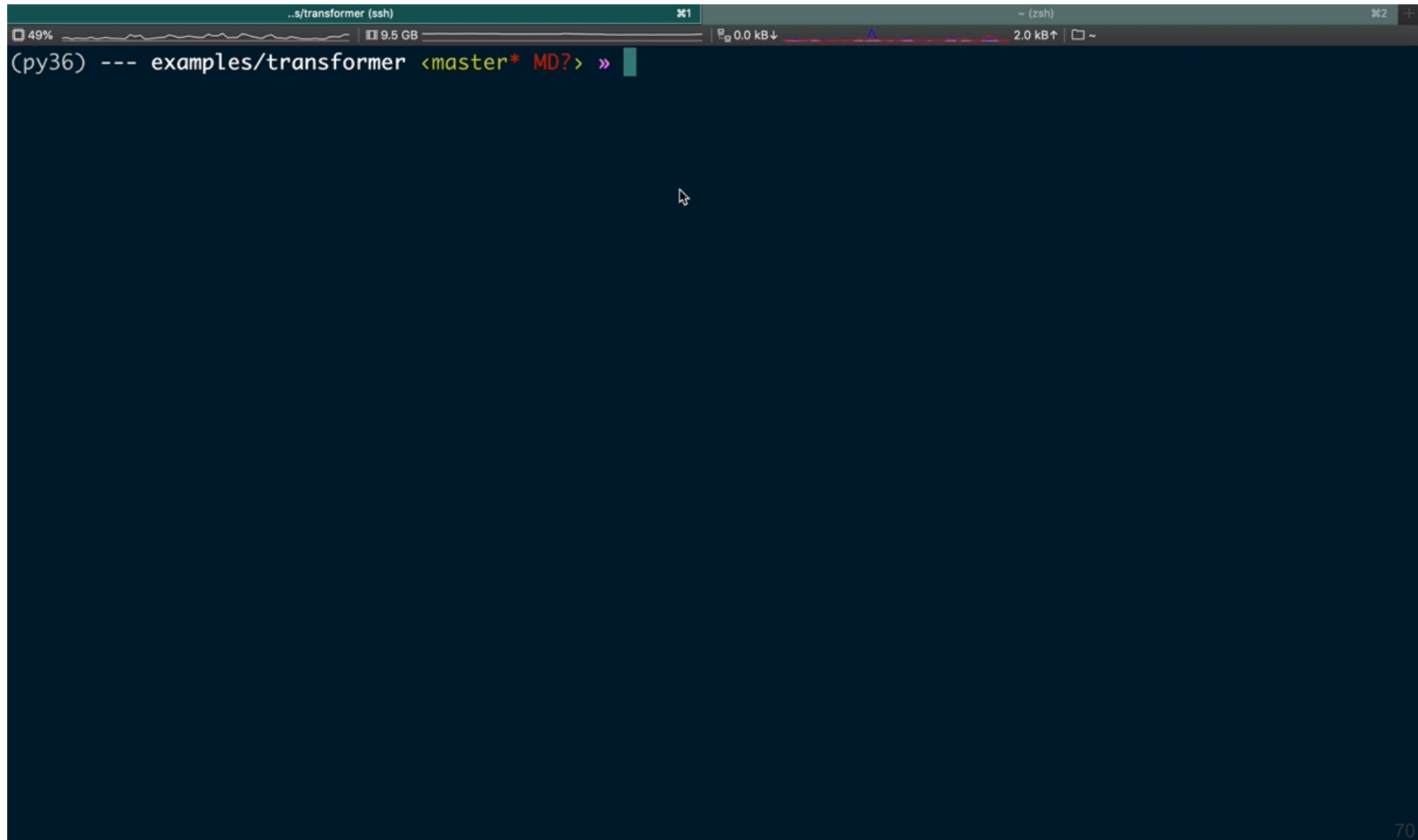


Running Example: MT - Learning from data instances

Data	{	<pre> 1 # Read data 2 dataset = PairedTextData(data_hparams) 3 batch = Dataloader(dataset).get_next() 4 # Encode 5 embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams) 6 encoder = TransformerEncoder(hparams=encoder_hparams) 7 enc_outputs = encoder(embedder(batch['source_text_ids']), 8 batch['source_length']) 9 # Build decoder 10 decoder = AttentionRNNDecoder(memory=enc_outputs, 11 hparams=decoder_hparams) 12 # Maximum Likelihood Estimation 13 ## Teacher-forcing decoding 14 outputs, length, _ = decoder(decoding_strategy='teacher-forcing', 15 inputs=embedder(batch['target_text_ids']), 16 seq_length=batch['target_length']-1) 17 ## Cross-entropy loss 18 loss = sequence_sparse_softmax_cross_entropy(19 labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length) 20 </pre>
Architecture & Inference	{	
Learning Loss	{	



DEMO



A terminal window with a dark blue background. The title bar shows "...s/transformer (ssh)" and "x1". The status bar at the top displays "49%", "9.5 GB", "0.0 kB ↓", and "2.0 kB ↑". The prompt is "(py36) --- examples/transformer <master* MD?> »" followed by a green cursor bar. A mouse cursor is visible in the center of the terminal area.

```
(py36) --- examples/transformer <master* MD?> »
```



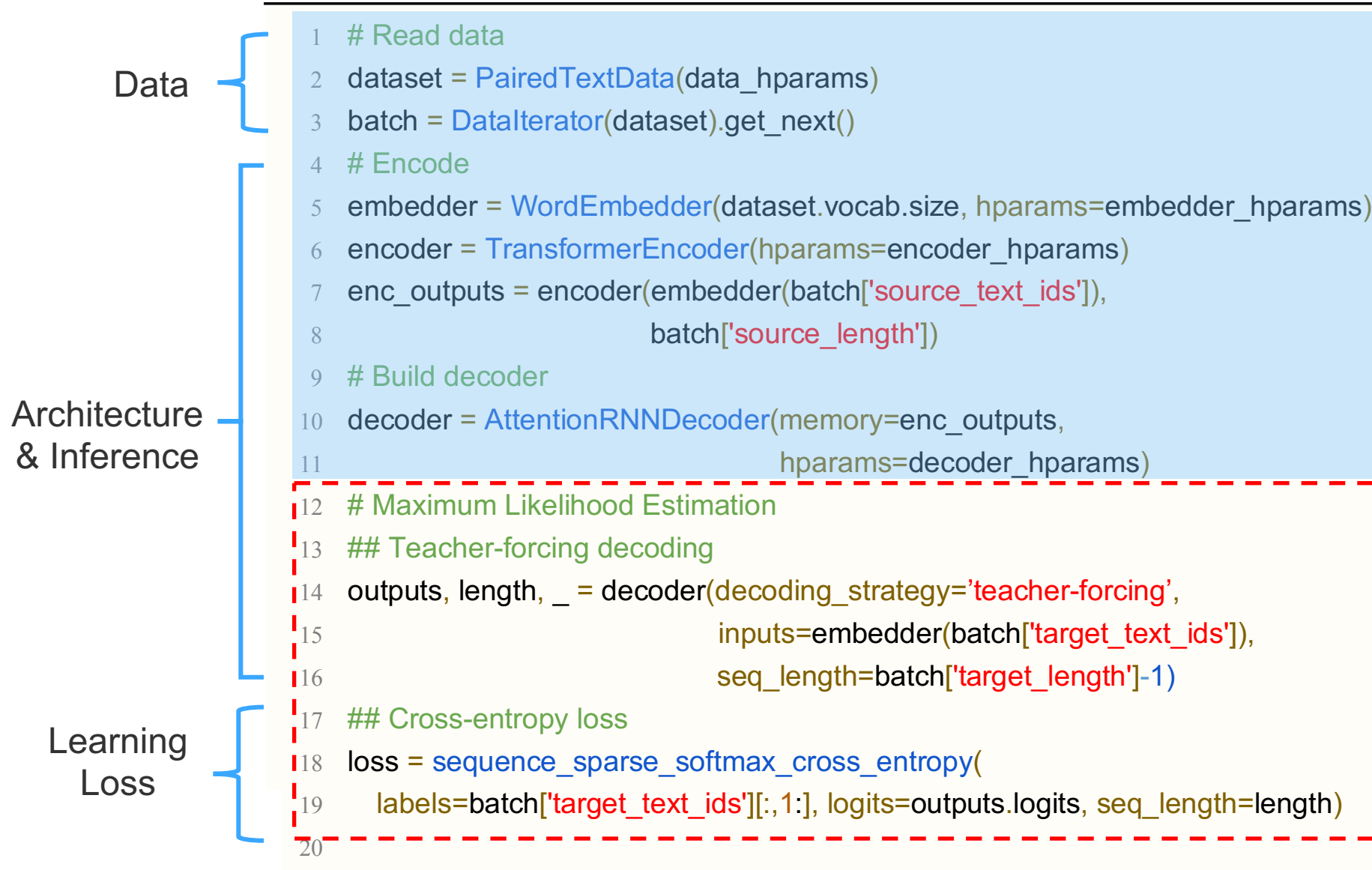
Running Example: MT - Learning from data instances

Data	{	<pre> 1 # Read data 2 dataset = PairedTextData(data_hparams) 3 batch = Dataloader(dataset).get_next() 4 # Encode 5 embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams) 6 encoder = TransformerEncoder(hparams=encoder_hparams) 7 enc_outputs = encoder(embedder(batch['source_text_ids']), 8 batch['source_length']) 9 # Build decoder 10 decoder = AttentionRNNDecoder(memory=enc_outputs, 11 hparams=decoder_hparams) </pre>
Architecture & Inference	{	<pre> 12 # Maximum Likelihood Estimation 13 ## Teacher-forcing decoding 14 outputs, length, _ = decoder(decoding_strategy='teacher-forcing', 15 inputs=embedder(batch['target_text_ids']), 16 seq_length=batch['target_length']-1) </pre>
Learning Loss	{	<pre> 17 ## Cross-entropy loss 18 loss = sequence_sparse_softmax_cross_entropy(19 labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length) 20 </pre>

**Maximum likelihood
Estimation**



Switching to using other experiences



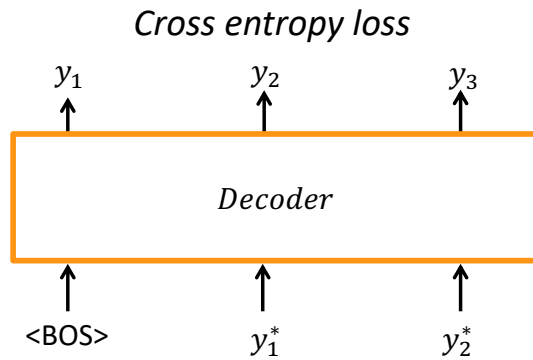
Keep unchanged

Maximum likelihood Estimation



Switching to using reward + data instances

- Maximum likelihood



```
# Teacher-forcing decoding
```

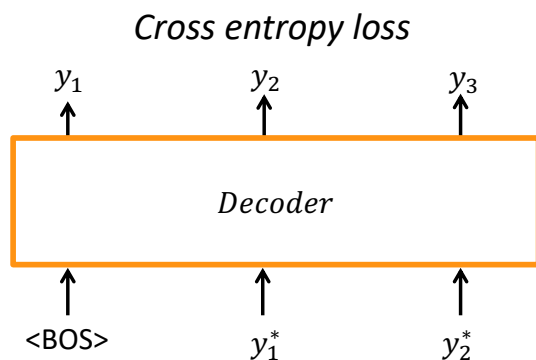
```
outputs, length, _ = decoder(decoding_strategy='teacher-forcing',  
                             inputs=embedder(batch['target_text_ids']),  
                             seq_length=batch['target_length']-1)
```

```
# Cross-entropy loss
```

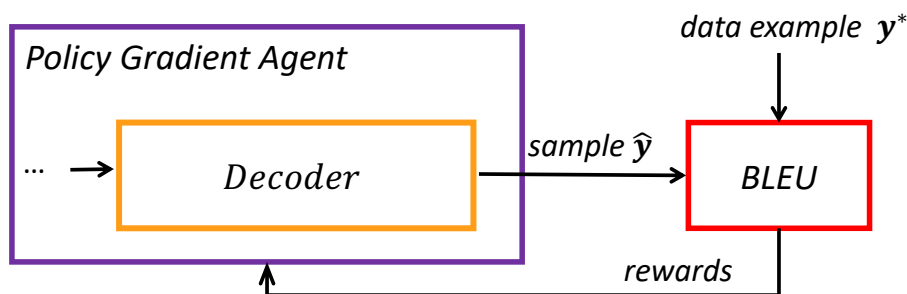
```
loss = sequence_sparse_softmax_cross_entropy(  
    labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length)
```

Switching to using reward + data instances

- Maximum likelihood



- Reinforcement learning



```
# Teacher-forcing decoding
```

```
outputs, length, _ = decoder(decoding_strategy='teacher-forcing',  
                             inputs=embedder(batch['target_text_ids']),  
                             seq_length=batch['target_length']-1)
```

```
# Cross-entropy loss
```

```
loss = sequence_sparse_softmax_cross_entropy(  
    labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length)
```

```
# Random sample decoding
```

```
outputs, length, _ = decoder(decoding_strategy='random_sample',  
                             start_tokens=[BOS]*batch_size, end_token=EOS,  
                             embedding=embedder)
```

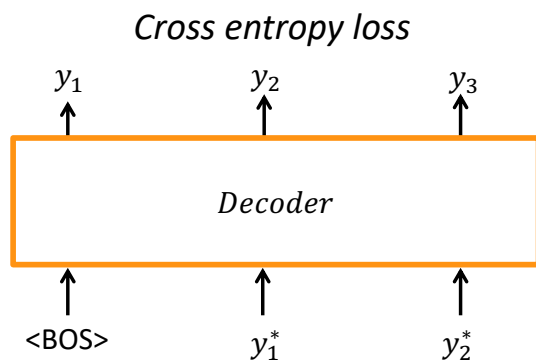
```
# Policy gradient agent for learning
```

```
agent = SeqPGAgent(  
    samples=outputs.sample_id, logits=outputs.logits, seq_length=length)
```

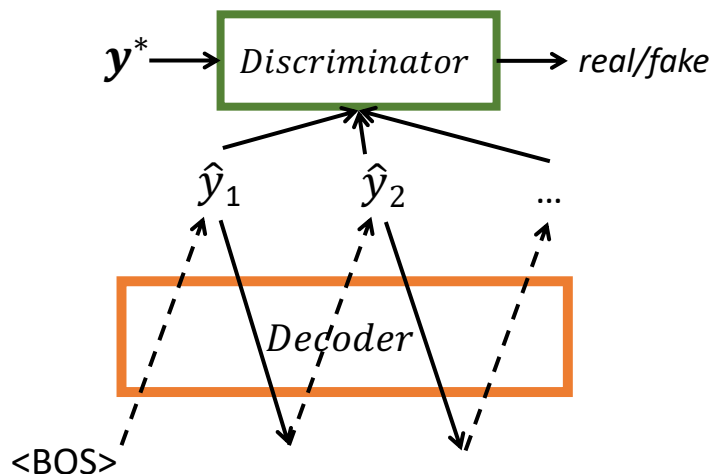
```
for _ in range(STEPS):  
    samples = agent.get_samples()  
    rewards = BLEU(batch['target_text_ids'], samples) # Reward  
    agent.observe(rewards)
```

Switching to using adversary + data instances

- Maximum likelihood



- Adversarial learning



```
# Teacher-forcing decoding
```

```
outputs, length, _ = decoder(decoding_strategy='teacher-forcing',  
                             inputs=embedder(batch['target_text_ids']),  
                             seq_length=batch['target_length']-1)
```

```
# Cross-entropy loss
```

```
loss = sequence_sparse_softmax_cross_entropy(  
    labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length)
```

```
# Gumbel-softmax decoding
```

```
outputs, _, _ = decoder(decoding_strategy='gumbel-softmax',  
                        start_tokens=[BOS]*batch_size, end_token=EOS,  
                        embedding=embedder)
```

```
discriminator = Conv1DClassifier(hparams=conv_hparams)
```

```
# Binary adversarial loss
```

```
G_loss, D_loss = binary_adversarial_losses(  
    embedder(batch['target_text_ids'][:, 1:]),  
    embedder(soft_ids=softmax(outputs.logits)),  
    discriminator)
```

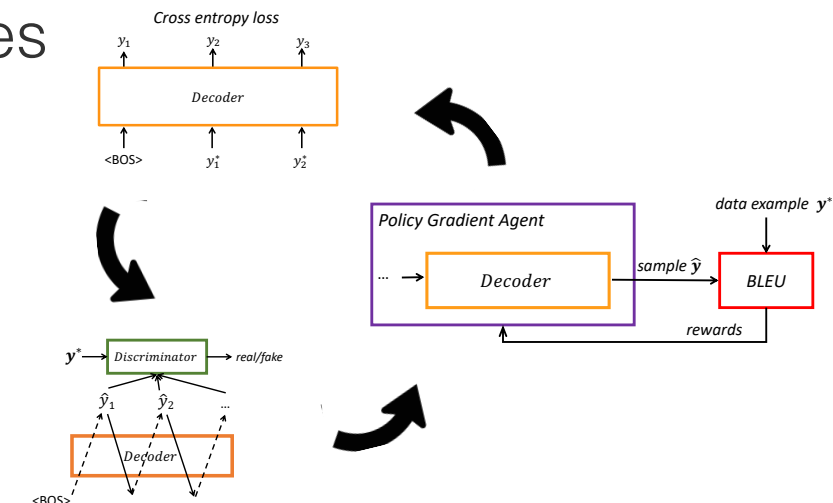
Summary of MT in Texar

- Highly modularized programming
 - Experience, arch, operation, loss, optimization
 - Intuitive conceptual-level APIs
- Easy switch between different forms of experiences
 - Plug in & out modules
 - No changes to irrelevant parts

```

1 # Read data
2 dataset = PairedTextData(data_hparams)
3 batch = Dataloader(dataset).get_next()
4 # Encode
5 embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams)
6 encoder = TransformerEncoder(hparams=encoder_hparams)
7 enc_outputs = encoder(embedder(batch['source_text_ids']),
8                       batch['source_length'])
9 # Build decoder
10 decoder = AttentionRNNDecoder(memory=enc_outputs,
11                              hparams=decoder_hparams)
12 # Maximum Likelihood Estimation
13 ## Teacher-forcing decoding
14 outputs, length, _ = decoder(decoding_strategy='teacher-forcing',
15                             inputs=embedder(batch['target_text_ids']),
16                             seq_length=batch['target_length']-1)
17 ## Cross-entropy loss
18 loss = sequence_sparse_softmax_cross_entropy(
19     labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length)
20

```

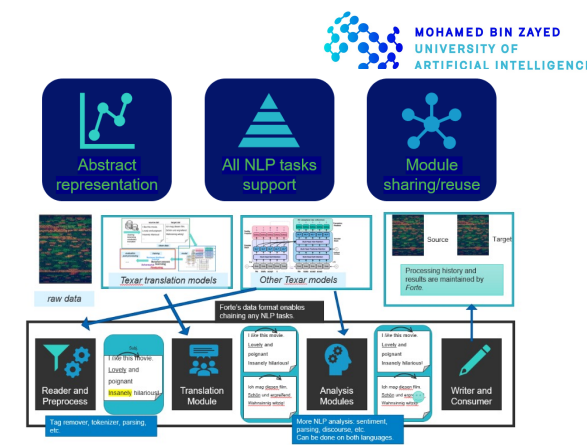


Applications of Texar

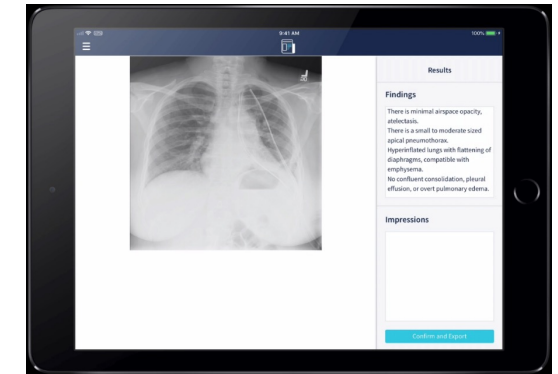
Many products built on Texar

- ❑ FORTE – templates for larger complex NLP applications
- ❑ Chest X-Ray report writer
- ❑ Medical Registry report writer
- ❑ ICD coding system
- ❑ Financial knowledge base builder
- ❑ Financial summary/report writer
- ❑ Multi-Lingual Cognitive Chat Bots
 - ❑ For Call Center Support
 - ❑ For Retail In-Store Assistance

FORTE



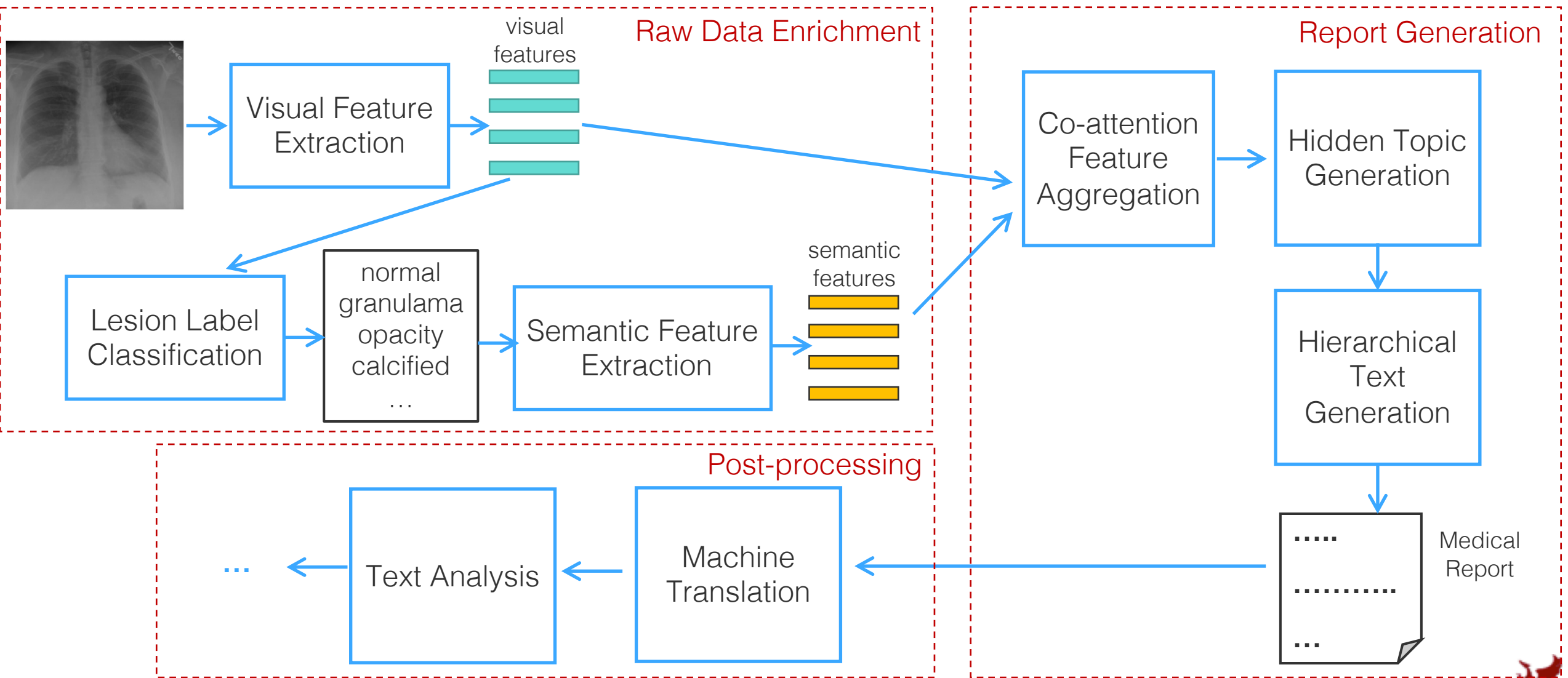
Chest X-Ray
Report Writer



Multi-Lingual
Cognitive
Chat Bots



Example: Chest X-Ray Report Writer



Example: Chest X-Ray Report Writer + Translation

无胸部X光片
拖拽至此上传胸部X光片(.png)

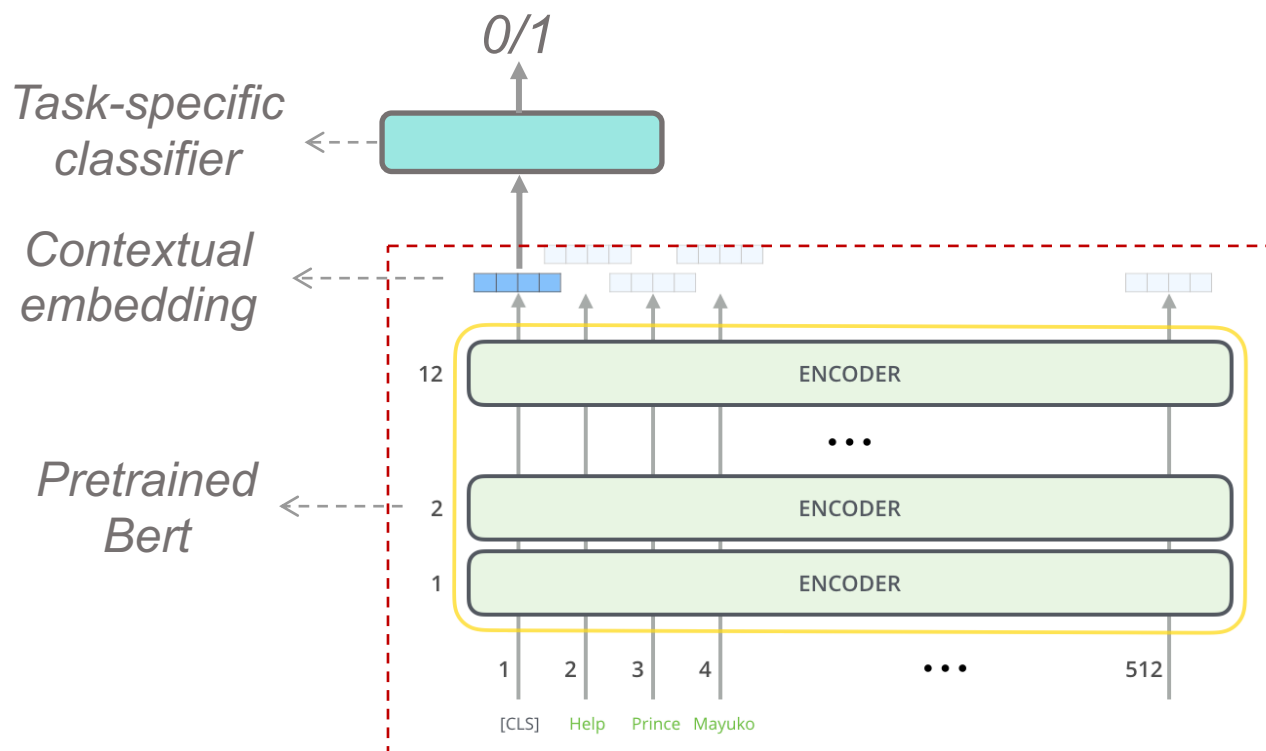
上传照片





Transfer Learning with Large Pretrained Models Using Texar

- Large neural models pretrained on massive data
 - Elmo, Bert, GPT-2, GPT-3, ...
- Pretrained models serve as building blocks to construct downstream models
 - Fine-tune using task-specific experiences



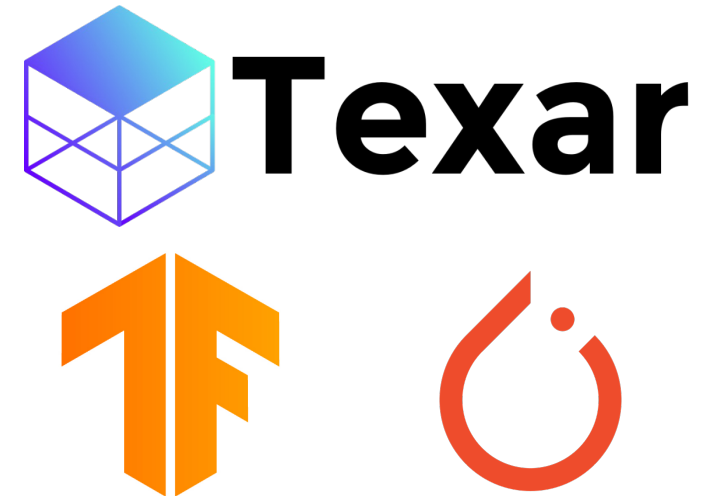
```
model = BertEncoder()  
_, features = model(input_ids, input_length, segment_ids)  
task_classifier = Conv1DClassifier(hparams={...})  
logits, preds = task_classifier(features[:,1,:])
```

OR

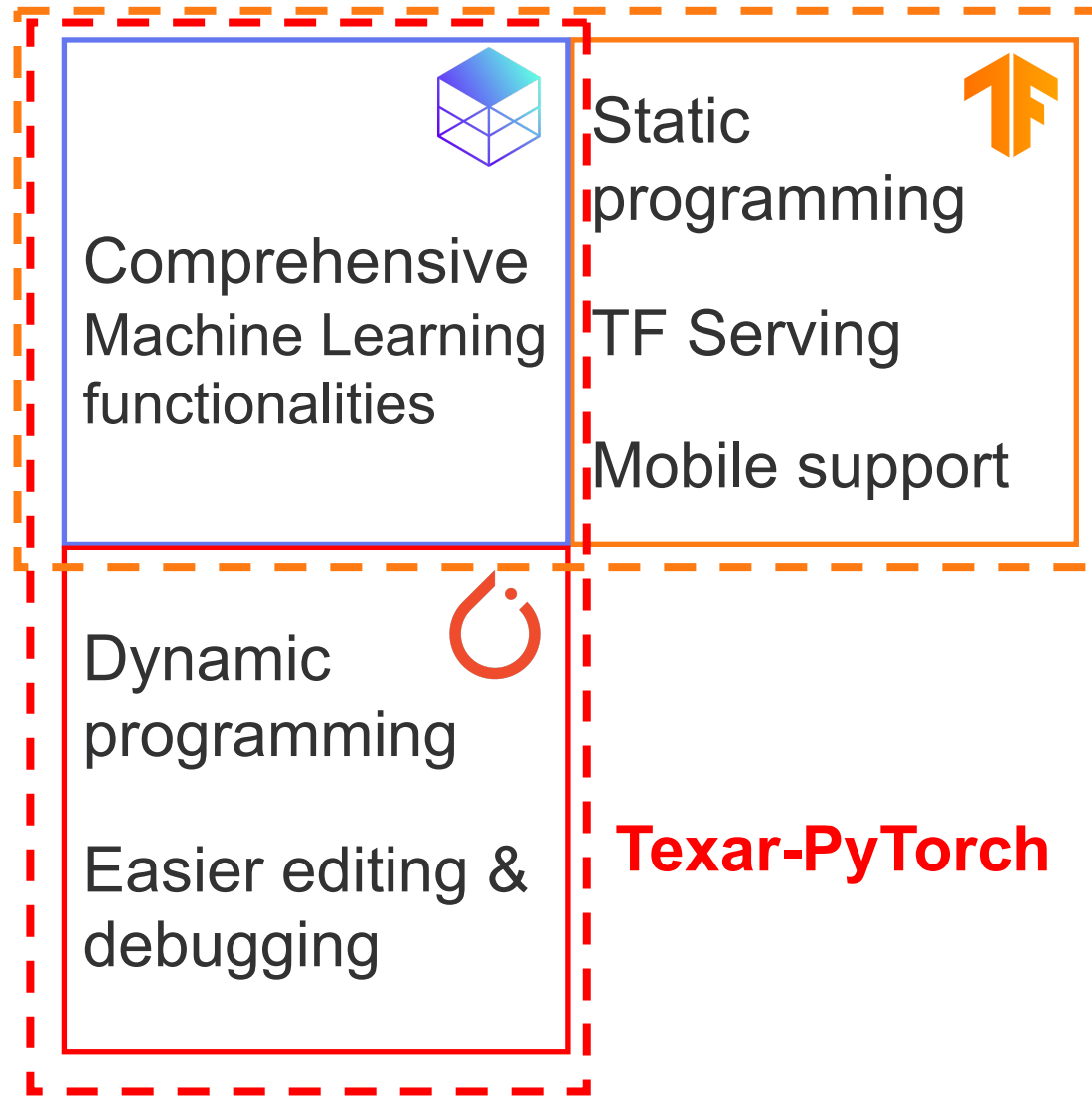
```
task_classifier = BertClassifier(hparams={'clas_strategy': 'cls_time'})  
logits, preds = task_classifier(input_ids, input_length, segment_ids)
```

Support of TensorFlow and PyTorch

- Texar is built upon TF and PyTorch
 - Texar-TF & Texar-PyTorch: mostly the same interfaces!
 - Higher-level intuitive APIs without loss of flexibility
 - Lots of ML components ready to use
- Combine the best design of TF and PyTorch
 - TF:
 - Easy and efficient data processing APIs
 - Excellent factorization of ML modules
 - Turnkey model training processor
 - PyTorch:
 - Intuitive programming interfaces
 - Transparent variable scope and sharing to users



Support of TensorFlow and PyTorch



Texar-TF

- Texar provides the same comprehensive supports of ML functionalities
- Fully compatible with native TF & PyTorch APIs
 - Get all other useful features of TF or PyTorch with **Texar-TF** & **Texar-PyTorch**



Texar Resources



Texar-TF



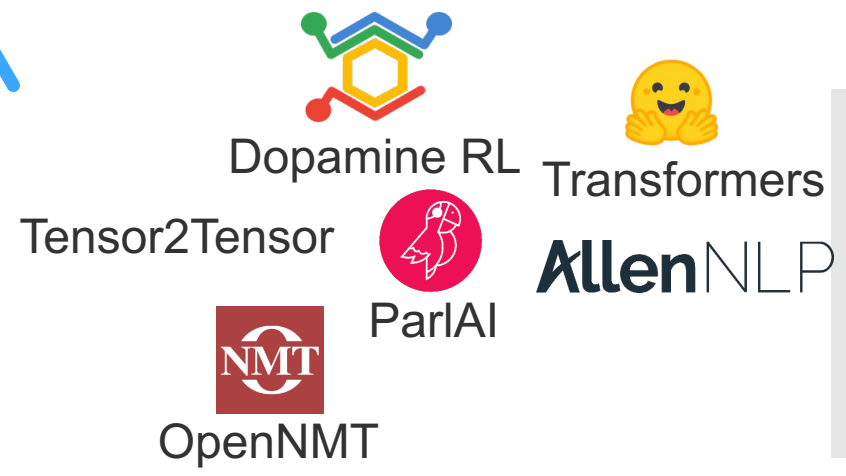
Texar-PyTorch

- Website: <https://asym1.io>
- GitHub (TF version): <https://github.com/asym1/texar>
- GitHub (PyTorch version): <https://github.com/asym1/texar-pytorch>
- Examples: <https://github.com/asym1/texar/blob/master/examples>
- Documentation: <https://texar.readthedocs.io/>
- Blog: <https://medium.com/@texar>
- Tech report: <https://arxiv.org/pdf/1809.00794.pdf>

Spectrum of Existing Tools

*domain-specific,
higher-level APIs*

*general,
lower-level APIs*



Interfaces at multiple abstraction levels

- Simplified APIs for common functionalities
- Advanced APIs for advanced functionalities and customizability



*Fixed Structure
Limited composability*

*Modularized,
Composable*



Spectrum of Existing Tools

- Symbolic languages for low-level development of ML models
 - Tensorflow, PyTorch, Keras, Jax, ...
- High-level, domain-specific APIs
 - Transformers: rich SOTA pretrained transformer models
 - AllenNLP: ready-to-use models for many NLP tasks
 - Dopamine RL: APIs for reinforcement learning
 - ...

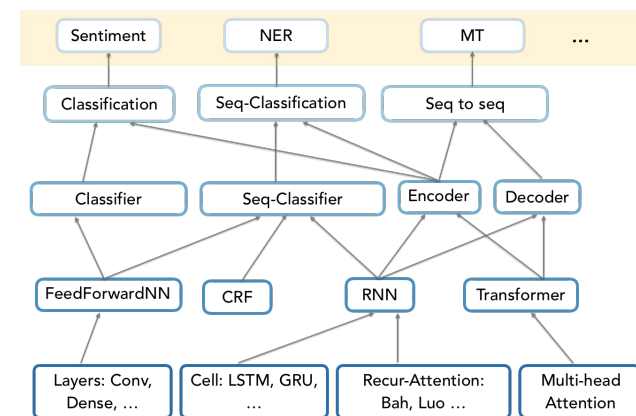
Summary of Part-II

- ML solution design
 - Plugging arbitrary available experiences in learning

$$\min_{q, \theta} -\mathbb{E} + \mathbb{D} - \mathbb{H}$$

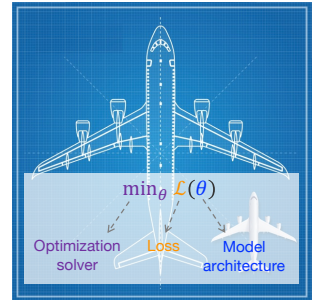
$$f = w_1 \cdot f(x | \text{📄}) + w_2 \cdot f(x | \text{📖}) + w_3 \cdot f(x | \text{💰}) + w_4 \cdot f(x | \text{📧}) + \dots$$

- Tooling for Compositionality in ML
 - Interfacing and composition between architecture, operation, loss (experience, divergence), optimization
 - ML compositionality with Texar



Schedule

- Lecture#1:** Theory: The Standard Model of ML
 A blueprint of ML paradigms for ALL experience
(Jan 19 Thursday, 4:45pm-6:15pm UK Time)
- Lecture#2:** Tooling: Operationalizing The Standard Model
 Compose your ML solutions like playing Lego
(Jan 20 Thursday, 1:00pm-2:30pm)
- Lecture#3:** Computing: Modern infrastructure for productive ML
 Automatic tuning, distributing, and scheduling
(Jan 20 Thursday, 4:45pm-6:15pm)



Petuum's open-source collection:
<https://github.com/petuum>

Summary

• Theory: The Standard Equation

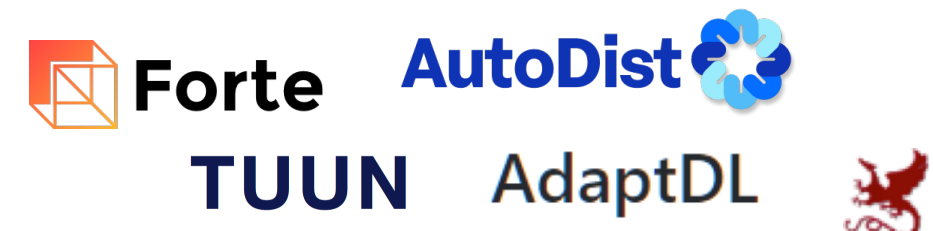
- A blueprint of ML paradigms for ALL experiences $\min_{q, \theta} - \mathbb{E}_{q(x, y)} [f(x, y)] + \alpha \mathbb{D}(q(x, y), p_{\theta}(x, y)) - \beta \mathbb{H}(q)$
- Experience function f can encode different types of experiences
 - Data instances, constraints, informativeness, reward, adversary models, ...
- Turnkey mechanism for learning with ALL experiences
 - Re-use originally specialized algorithms to other contexts
 - Experience compositional

• Tooling: Operationalizing “Learning with all experiences”

- ML solution design by plugging arbitrary experiences in learning
- ML compositionality with  **Texar** $f = w_1 \cdot f(x | \text{📄}) + w_2 \cdot f(x | \text{📖}) + w_3 \cdot f(x | \text{💰}) + w_4 \cdot f(x | \text{🏠}) + \dots$

• Computing: Modern infrastructure for productive ML

- Task/model interoperation
- Automatic tuning, distributing, and scheduling



Food for thoughts: How far would this take us?

- Physics

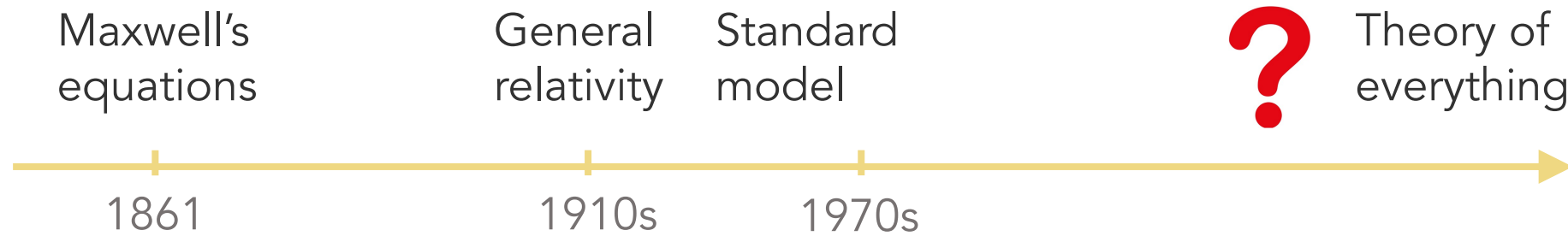


*It is only slightly overstating the case to say that **physics is the study of symmetry.***

-- Phil Anderson (1923-2020), Physicist, Nobel laureate

Food for thoughts: How far would this take us?

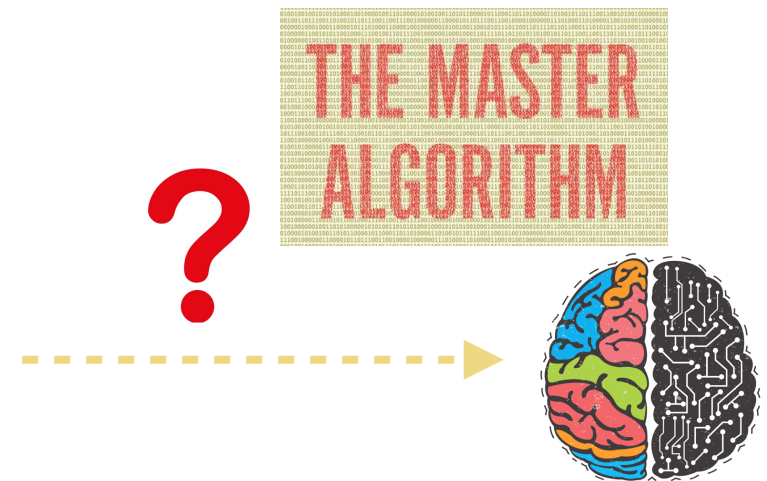
- Physics



- Machine Learning

Unified way of thinking

- ◆ Systematic understanding
- ◆ Automated solution creation
- ◆ Improved ML productivity and accessibility



References

- [1] Jun Zhu, Ning Chen, and Eric P Xing. 2014. Bayesian inference with posterior regularization and applications to infinite latent SVMs. JMLR(2014).
- [2] Jun Zhu and Eric P Xing. 2009. Maximum Entropy Discrimination Markov Networks. JMLR(2009).
- [3] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In ACL.
- [4] Zhiting Hu, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He, Lianhui Qin, Di Wang, Xuezhe Ma, et al. 2019. Texar: A modularized, versatile, and extensible toolkit for text generation. ACL(2019).
- [5] Zhiting Hu, Bowen Tan, Russ R Salakhutdinov, Tom M Mitchell, and Eric P Xing. 2019. Learning data manipulation for augmentation and weighting. In NeurIPS.
- [6] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In ICML.
- [7] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P Xing. 2018. On Unifying Deep Generative Models. In ICLR.
- [8] Zhiting Hu, Zichao Yang, Russ R Salakhutdinov, Lianhui Qin, Xiaodan Liang, Haoye Dong, and Eric P Xing. 2018. Deep generative models with learnable knowledge constraints. In NeurIPS.

References

- [9] Hao Zhang, Zeyu Zheng, Shizhen Xu, Wei Dai, Qirong Ho, Xiaodan Liang, Zhiting Hu, Jinliang Wei, Pengtao Xie, Eric Xing. Poseidon: An Efficient Communication Architecture for Distributed Deep Learning on GPU Clusters. ATC 2017
- [10] Wei Dai, Yi Zhou, Nanqing Dong, Hao Zhang, Eric P. Xing. Toward Understanding the Impact of Staleness in Distributed Machine Learning. ICLR 2019
- [11] Hao Zhang*, Shizhen Xu*, Graham Neubig, Wei Dai, Qirong Ho, Guangwen Yang, and Eric P. Xing. Cavs: A Vertex-centric Programming Interface for Dynamic Neural Networks. ATC 2018
- [12] Aurick Qiao, Abutalib Aghayev, Weiren Yu, Haoyang Chen, Qirong Ho, Garth A Gibson, Eric P Xing. Litz: Elastic Framework for High-Performance Distributed Machine Learning. ATC 2018
- [13] Kirthivasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R Collins, Jeff Schneider, Barnabas Poczos, Eric P Xing. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. JMLR 2020
- [14] E. P. Xing, Q. Ho, P. Xie, W. Dai, Strategies and Principles of Distributed Machine Learning on Big Data, Engineering, Volume:2, pp179 - 95, 2016.

UC San Diego



Carnegie Mellon University
School of Computer Science

Petuum

Thank You