

Definition Learning

FutureAI Final Project

Colin McDonnell

how humans learn

definition

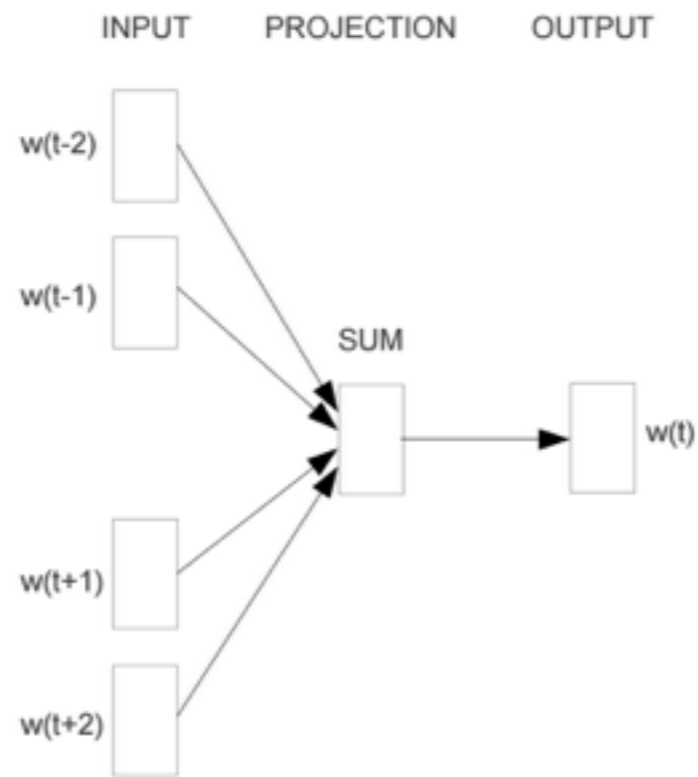
see a few usage examples

how machines learn

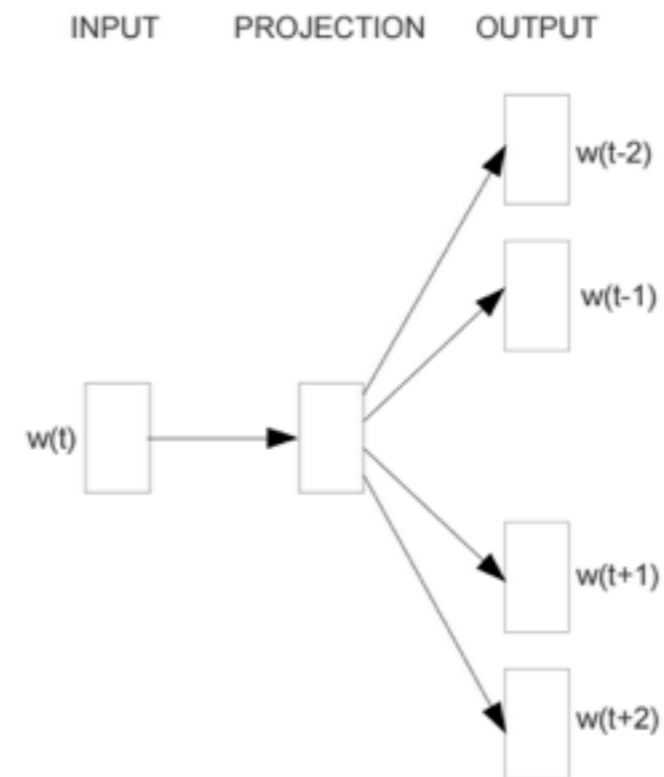
prediction

see a few hundred million usage examples

how machines learn



CBOW



Skip-gram

how machines learn

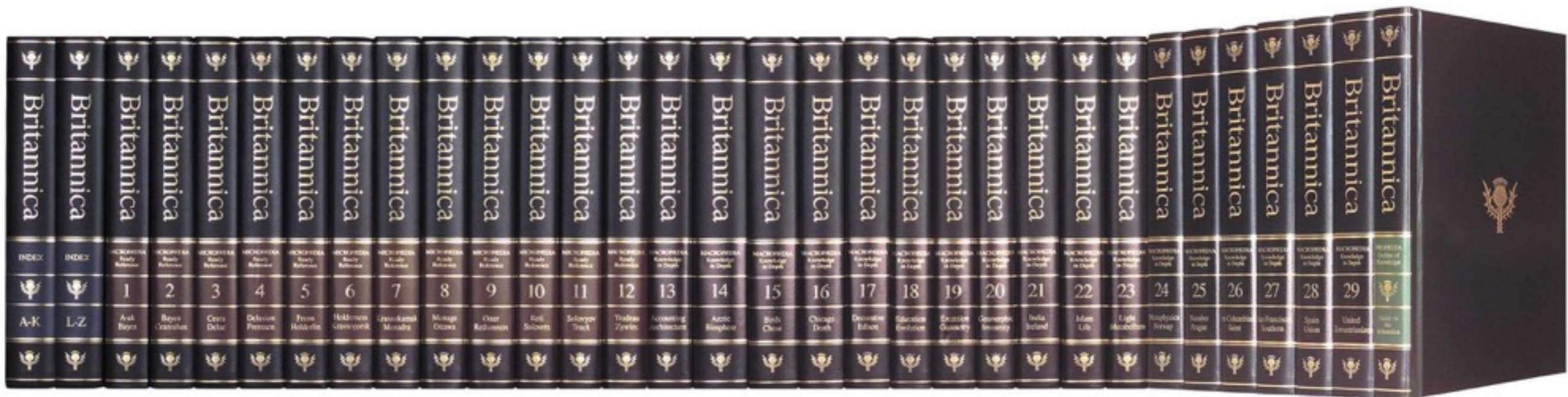
Billion Words Corpus



x2000

how machines learn

Billion Words Corpus



x23

goal

teach machines to learn like us

approach

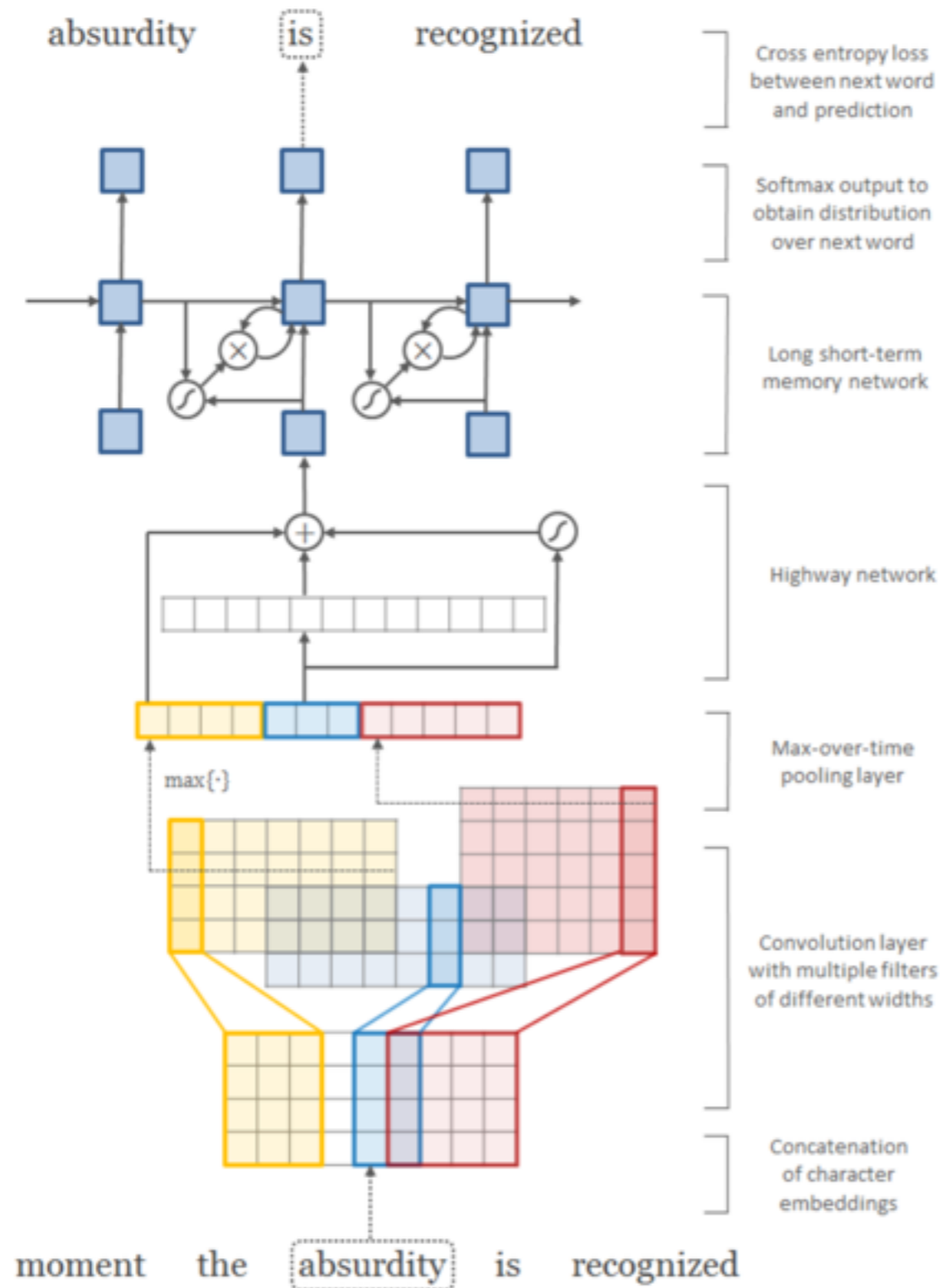
download Webster's Dictionary

```
{  defn : ["extending", "far", "down", "from", "the", "top", "or", "surface"],  
  word : "deep" }
```

pass definition words into network one at a time

final activation should be close to true word embedding

how to handle variable length inputs

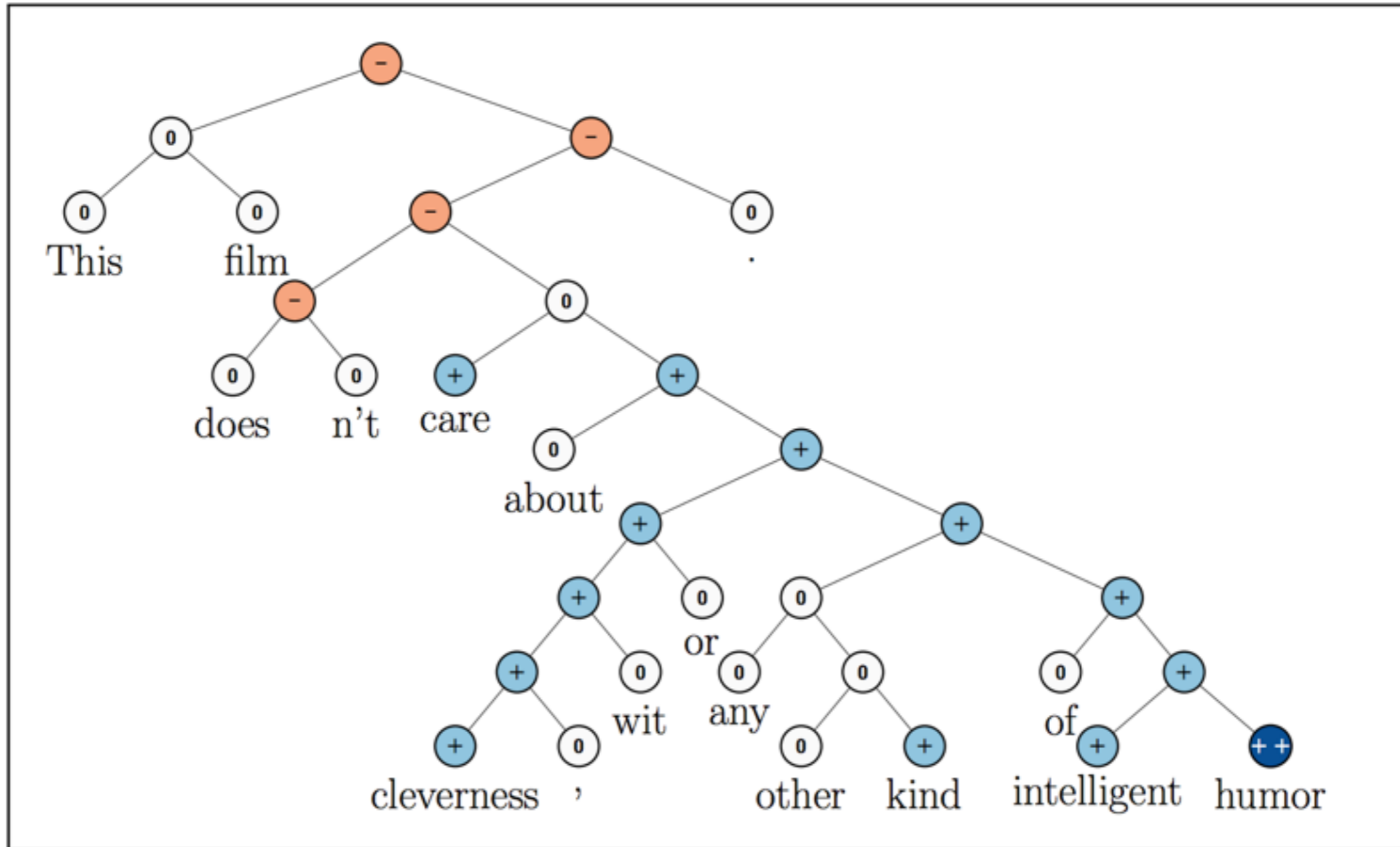


Character-Aware Neural Language Models

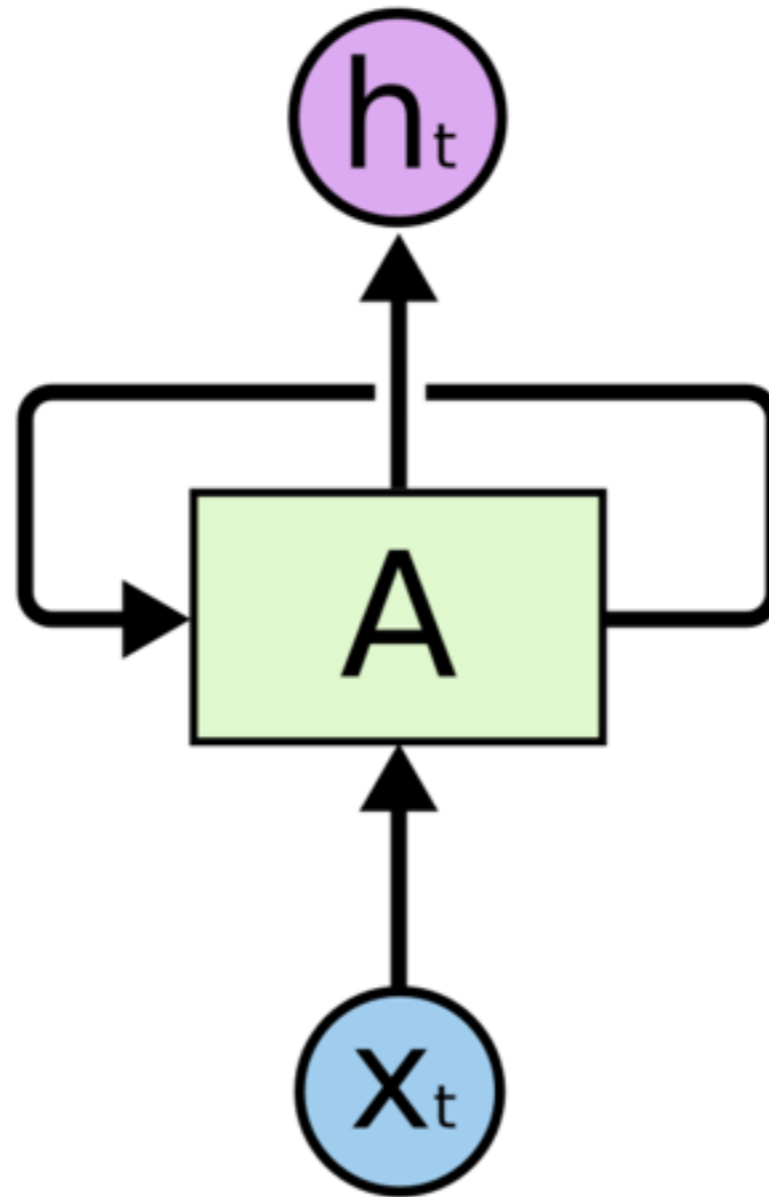
Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush

(Submitted on 26 Aug 2015 (v1), last revised 1 Dec 2015 (this version, v4))

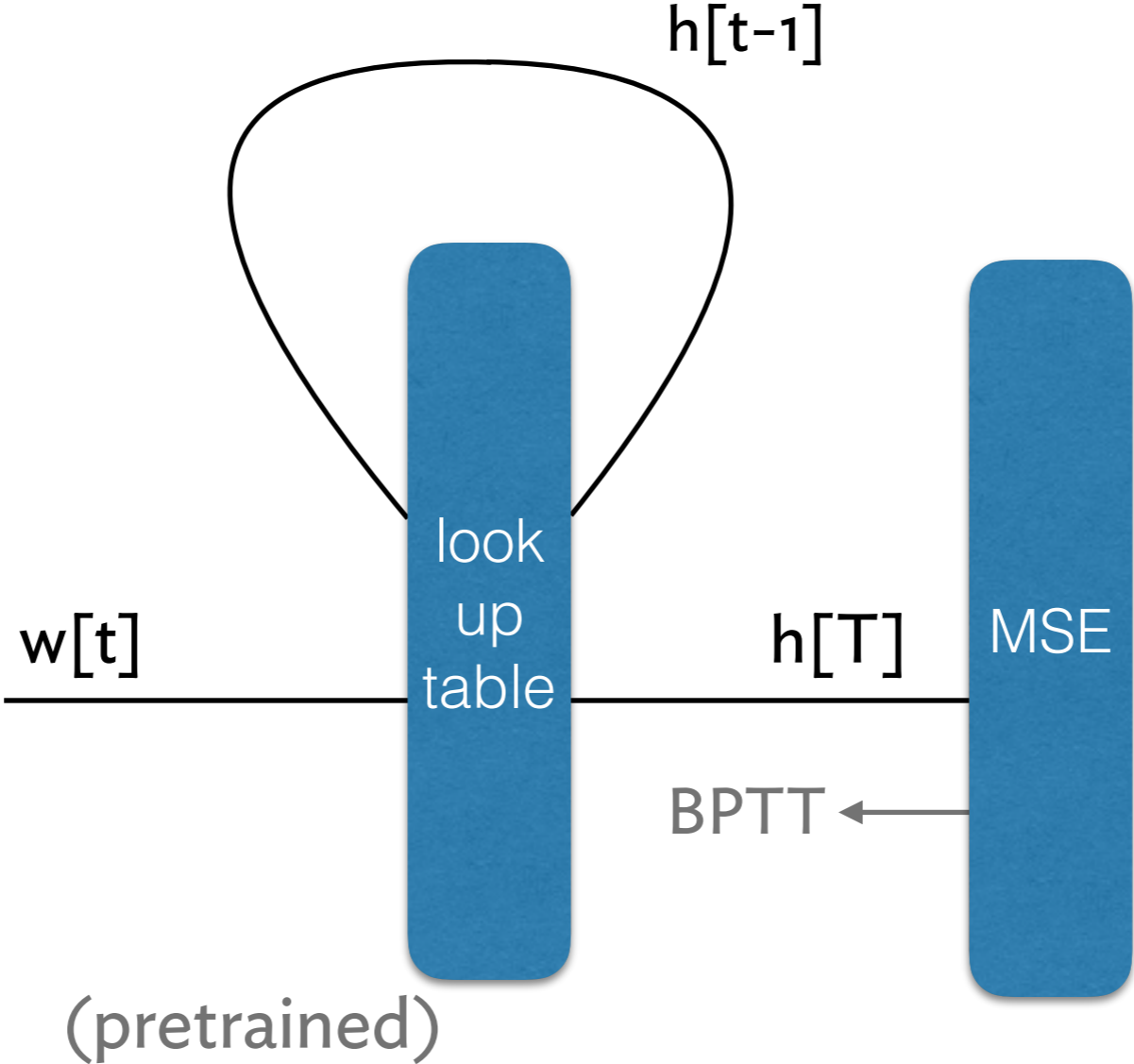
how to handle variable length inputs



how to handle variable length inputs



architecture



results

days of continuous failure and misery

results

```
...
113  def main(argv):
114  """
115  Main function
116  """
117  # Parse command-line arguments
118  parser = argparse.ArgumentParser()
119  parser.add_argument("-i", "--input", type=str, help="Input file")
120  parser.add_argument("-o", "--output", type=str, help="Output file")
121  parser.add_argument("-v", "--verbose", action="store_true", help="Verbose output")
122  args = parser.parse_args()
123
124  # Load input data
125  input_data = load_data(args.input)
126
127  # Process data
128  result = process_data(input_data, args.verbose)
129
130  # Save output data
131  save_data(result, args.output)
132
133  if __name__ == "__main__":
134  main(sys.argv)
```

```
...
135  def load_data(filename):
136  """
137  Load input data from a file
138  """
139  data = []
140  with open(filename, "r") as f:
141  for line in f:
142  line = line.strip()
143  if line:
144  data.append(line)
145  return data
146
147  def process_data(data, verbose):
148  """
149  Process input data
150  """
151  result = []
152  for i, line in enumerate(data):
153  # Parse line
154  parsed_line = parse_line(line)
155
156  # Process parsed line
157  processed_line = process_parsed_line(parsed_line, verbose)
158
159  # Append to result
160  result.append(processed_line)
161
162  return result
163
164  def save_data(data, filename):
165  """
166  Save output data to a file
167  """
168  with open(filename, "w") as f:
169  for line in data:
170  f.write(line + "\n")
171
172  def parse_line(line):
173  """
174  Parse a line of input data
175  """
176  # Split line into fields
177  fields = line.split(",")
178
179  # Parse fields
180  parsed_line = {}
181  for i, field in enumerate(fields):
182  # Parse field
183  value = parse_field(field, i)
184
185  # Add to parsed line
186  parsed_line["field_{}".format(i)] = value
187
188  return parsed_line
189
190  def process_parsed_line(parsed_line, verbose):
191  """
192  Process a parsed line of input data
193  """
194  # Extract fields from parsed line
195  fields = {}
196  for key, value in parsed_line.items():
197  fields[key] = value
198
199  # Process fields
200  processed_line = {}
201  for key, value in fields.items():
202  # Process field
203  result = process_field(key, value, verbose)
204
205  # Add to processed line
206  processed_line[key] = result
207
208  return processed_line
209
210  def process_field(key, value, verbose):
211  """
212  Process a field of input data
213  """
214  # Process field
215  result = value
216
217  # Return result
218  return result
219
220  def parse_field(field, i):
221  """
222  Parse a field of input data
223  """
224  # Parse field
225  value = field
226
227  # Return value
228  return value
```

```
...
229  def process_data(input_data, verbose):
230  """
231  Process input data
232  """
233  result = []
234  for i, line in enumerate(input_data):
235  # Parse line
236  parsed_line = parse_line(line)
237
238  # Process parsed line
239  processed_line = process_parsed_line(parsed_line, verbose)
240
241  # Append to result
242  result.append(processed_line)
243
244  return result
245
246  def save_data(data, filename):
247  """
248  Save output data to a file
249  """
250  with open(filename, "w") as f:
251  for line in data:
252  f.write(line + "\n")
253
254  def parse_line(line):
255  """
256  Parse a line of input data
257  """
258  # Split line into fields
259  fields = line.split(",")
260
261  # Parse fields
262  parsed_line = {}
263  for i, field in enumerate(fields):
264  # Parse field
265  value = parse_field(field, i)
266
267  # Add to parsed line
268  parsed_line["field_{}".format(i)] = value
269
270  return parsed_line
271
272  def process_parsed_line(parsed_line, verbose):
273  """
274  Process a parsed line of input data
275  """
276  # Extract fields from parsed line
277  fields = {}
278  for key, value in parsed_line.items():
279  fields[key] = value
280
281  # Process fields
282  processed_line = {}
283  for key, value in fields.items():
284  # Process field
285  result = process_field(key, value, verbose)
286
287  # Add to processed line
288  processed_line[key] = result
289
290  return processed_line
291
292  def process_field(key, value, verbose):
293  """
294  Process a field of input data
295  """
296  # Process field
297  result = value
298
299  # Return result
300  return result
301
302  def parse_field(field, i):
303  """
304  Parse a field of input data
305  """
306  # Parse field
307  value = field
308
309  # Return value
310  return value
```

```
...
311  def parse_line(line):
312  """
313  Parse a line of input data
314  """
315  # Split line into fields
316  fields = line.split(",")
317
318  # Parse fields
319  parsed_line = {}
320  for i, field in enumerate(fields):
321  # Parse field
322  value = parse_field(field, i)
323
324  # Add to parsed line
325  parsed_line["field_{}".format(i)] = value
326
327  return parsed_line
328
329  def process_parsed_line(parsed_line, verbose):
330  """
331  Process a parsed line of input data
332  """
333  # Extract fields from parsed line
334  fields = {}
335  for key, value in parsed_line.items():
336  fields[key] = value
337
338  # Process fields
339  processed_line = {}
340  for key, value in fields.items():
341  # Process field
342  result = process_field(key, value, verbose)
343
344  # Add to processed line
345  processed_line[key] = result
346
347  return processed_line
348
349  def process_field(key, value, verbose):
350  """
351  Process a field of input data
352  """
353  # Process field
354  result = value
355
356  # Return result
357  return result
358
359  def parse_field(field, i):
360  """
361  Parse a field of input data
362  """
363  # Parse field
364  value = field
365
366  # Return value
367  return value
```

```
...
368  def parse_line(line):
369  """
370  Parse a line of input data
371  """
372  # Split line into fields
373  fields = line.split(",")
374
375  # Parse fields
376  parsed_line = {}
377  for i, field in enumerate(fields):
378  # Parse field
379  value = parse_field(field, i)
380
381  # Add to parsed line
382  parsed_line["field_{}".format(i)] = value
383
384  return parsed_line
385
386  def process_parsed_line(parsed_line, verbose):
387  """
388  Process a parsed line of input data
389  """
390  # Extract fields from parsed line
391  fields = {}
392  for key, value in parsed_line.items():
393  fields[key] = value
394
395  # Process fields
396  processed_line = {}
397  for key, value in fields.items():
398  # Process field
399  result = process_field(key, value, verbose)
400
401  # Add to processed line
402  processed_line[key] = result
403
404  return processed_line
405
406  def process_field(key, value, verbose):
407  """
408  Process a field of input data
409  """
410  # Process field
411  result = value
412
413  # Return result
414  return result
415
416  def parse_field(field, i):
417  """
418  Parse a field of input data
419  """
420  # Parse field
421  value = field
422
423  # Return value
424  return value
```

```
...
425  def parse_line(line):
426  """
427  Parse a line of input data
428  """
429  # Split line into fields
430  fields = line.split(",")
431
432  # Parse fields
433  parsed_line = {}
434  for i, field in enumerate(fields):
435  # Parse field
436  value = parse_field(field, i)
437
438  # Add to parsed line
439  parsed_line["field_{}".format(i)] = value
440
441  return parsed_line
442
443  def process_parsed_line(parsed_line, verbose):
444  """
445  Process a parsed line of input data
446  """
447  # Extract fields from parsed line
448  fields = {}
449  for key, value in parsed_line.items():
450  fields[key] = value
451
452  # Process fields
453  processed_line = {}
454  for key, value in fields.items():
455  # Process field
456  result = process_field(key, value, verbose)
457
458  # Add to processed line
459  processed_line[key] = result
460
461  return processed_line
462
463  def process_field(key, value, verbose):
464  """
465  Process a field of input data
466  """
467  # Process field
468  result = value
469
470  # Return result
471  return result
472
473  def parse_field(field, i):
474  """
475  Parse a field of input data
476  """
477  # Parse field
478  value = field
479
480  # Return value
481  return value
```

```
...
482  def parse_line(line):
483  """
484  Parse a line of input data
485  """
486  # Split line into fields
487  fields = line.split(",")
488
489  # Parse fields
490  parsed_line = {}
491  for i, field in enumerate(fields):
492  # Parse field
493  value = parse_field(field, i)
494
495  # Add to parsed line
496  parsed_line["field_{}".format(i)] = value
497
498  return parsed_line
499
500  def process_parsed_line(parsed_line, verbose):
501  """
502  Process a parsed line of input data
503  """
504  # Extract fields from parsed line
505  fields = {}
506  for key, value in parsed_line.items():
507  fields[key] = value
508
509  # Process fields
510  processed_line = {}
511  for key, value in fields.items():
512  # Process field
513  result = process_field(key, value, verbose)
514
515  # Add to processed line
516  processed_line[key] = result
517
518  return processed_line
519
520  def process_field(key, value, verbose):
521  """
522  Process a field of input data
523  """
524  # Process field
525  result = value
526
527  # Return result
528  return result
529
530  def parse_field(field, i):
531  """
532  Parse a field of input data
533  """
534  # Parse field
535  value = field
536
537  # Return value
538  return value
```

evaluation

train on half of the dataset

see mean squared error of predicted embeddings for words in test set

calculate perplexity

test embeddings on analogical reasoning tasks

future work

curriculum learning: order matters

DAGify the dictionary

find axiomatic subset of language

learn that subset in an unsupervised way or with innate biases

pair with “definition detector” for high-impact updates

Turing Test system

working memory

hierarchical SFA with increasing time constants

long-term memory

definition learning + omni-scale skip-gram

recurrent attention mechanism

over working memory

over long term memory: associational vector space

over internal output buffer (DRAW-style)

over decision to output

text generation from vector representation

train on SAT grammar questions, use as training signal

self-play for AI chatbots?