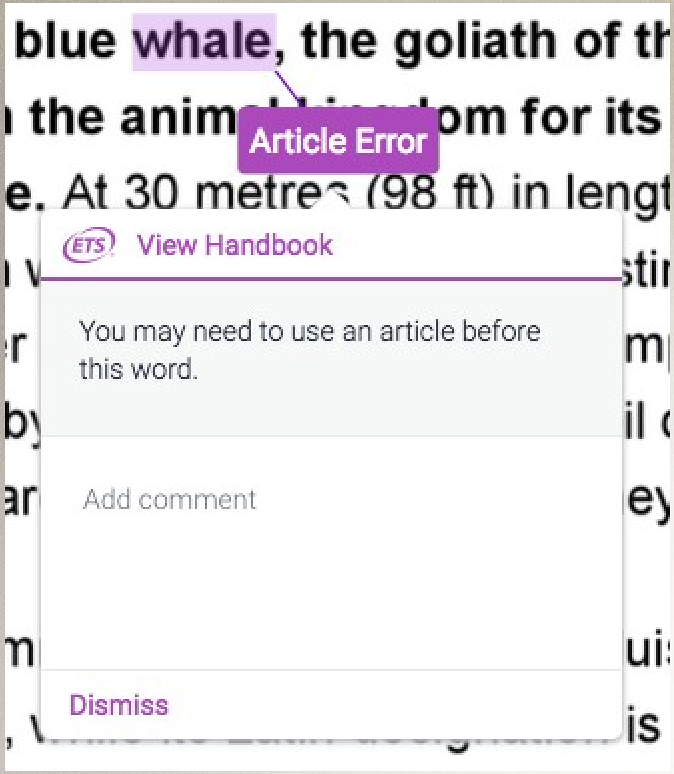
Automated Essay Scoring

Alexander Klapheke

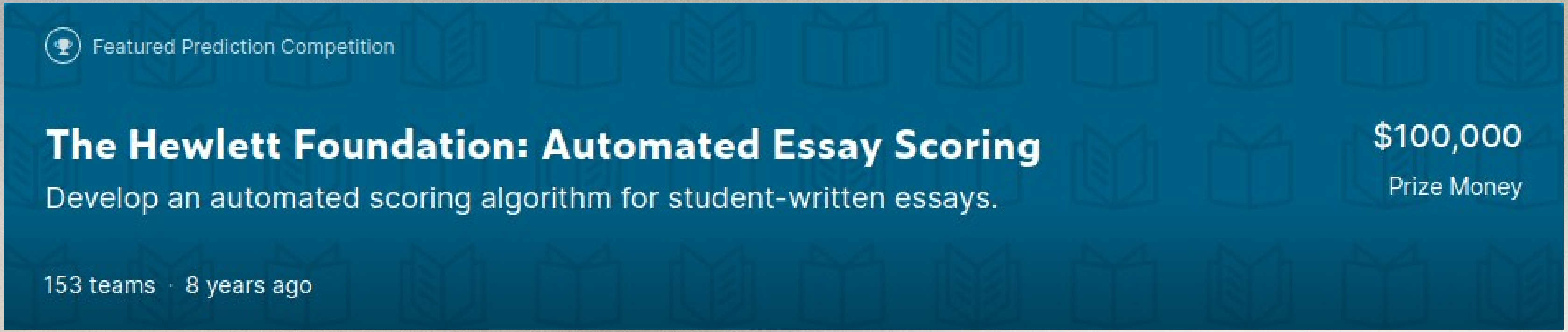
DSI-11 Capstone project

June 9, 2020

In 1999, ETS began using *e-rater*, which is now used to grade the SAT essay, along with a human reader.



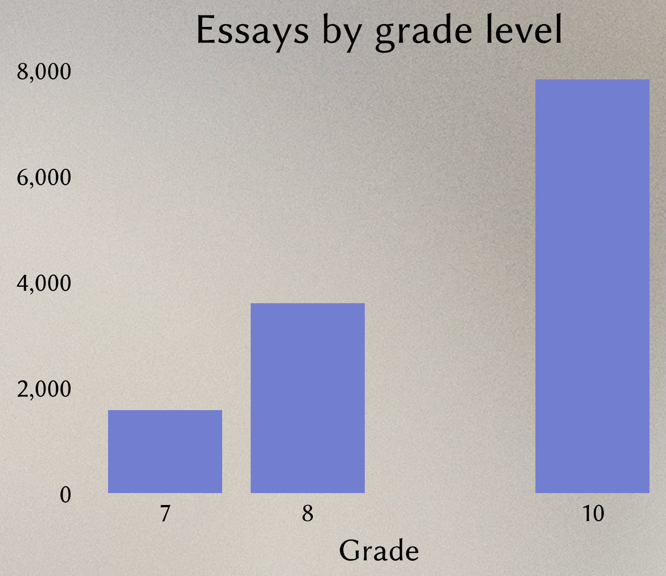
In 2012, the Hewlitt Foundation released ASAP,* a dataset of 13,000 middle- & high-school essays.

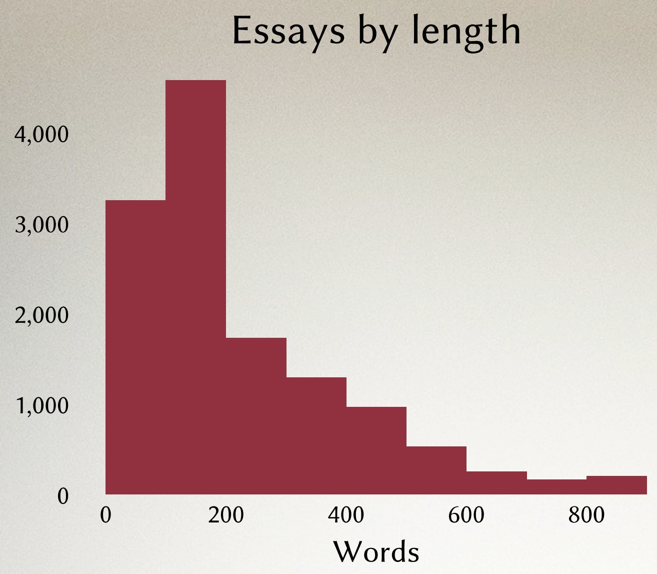


* Automated Student Assessment Prize

kaggle.com/c/asap-aes

In 2012, the Hewlitt Foundation released ASAP, a dataset of 13,000 middle- & high-school essays.





In 2012, the Hewlitt Foundation released ASAP, a dataset of 13,000 middle- & high-school essays.

Set	Grade level	Style	Score range
1	8	Persuasion	2–12
2	10	Persuasion	1–6, 1–4
3	10	Exposition	0–3
4	10	Exposition	0–3
5	8	Exposition	0–4
6	10	Exposition	0–4
7	7	Narrative	0–30
8	10	Narrative	0–60

My training set: the ~3,500 essays in sets 3 & 4, which are comparable but topically different.

Set	Grade level	Style	Score range
1	8	Persuasion	2–12
2	10	Persuasion	1–6, 1–4
3	10	Exposition	0–3
4	10	Exposition	0–3
5	8	Exposition	0–4
6	10	Exposition	0–4
7	7	Narrative	0–30
8	10	Narrative	0–60

We'd like an essay scorer to take into account:

Completion

Structure

Relevance

Grammar/usage/mechanics

The ETS claims its *e-rater* accounts for prompt-relevance, as well as:

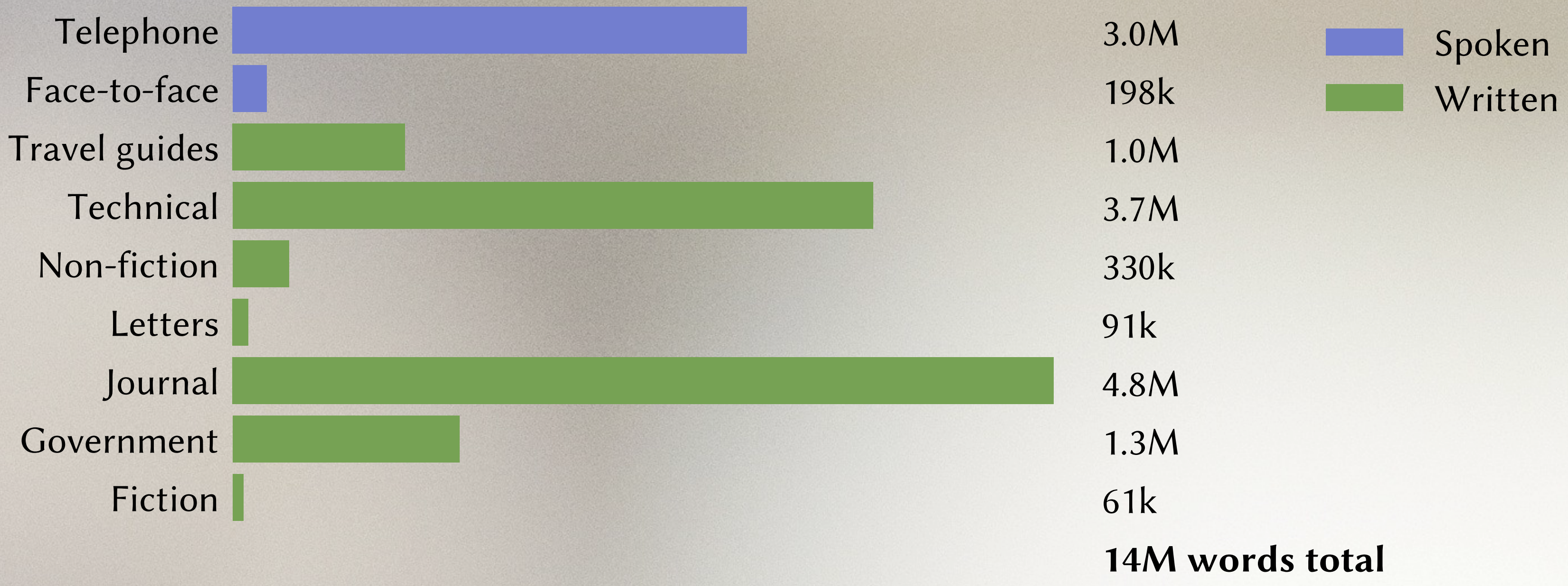
- errors in grammar (e.g., subject-verb agreement)
- usage (e.g., preposition selection)
- mechanics (e.g., capitalization)
- style (e.g., repetitious word use)
- discourse structure (e.g., presence of a thesis statement, main points)
- vocabulary usage (e.g., relative sophistication of vocabulary)
- sentence variety
- source use
- discourse coherence quality

ets.org/research/topics/as_nlp/writing_quality/

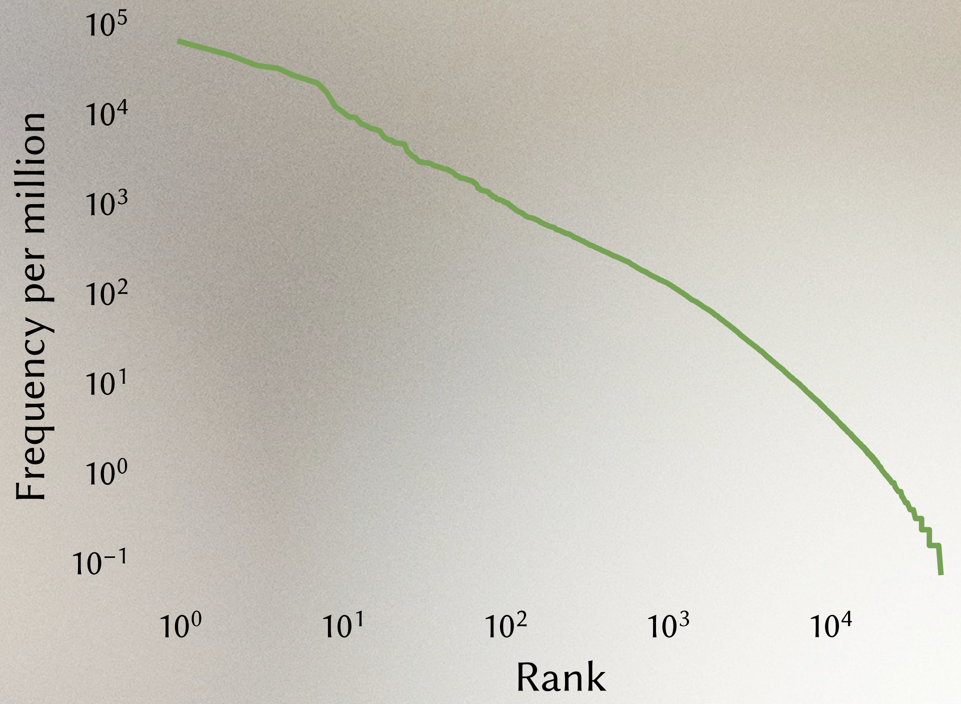
I compiled metrics to stand in for scoring criteria:

Completion	Token count (r = 0.72)
Vocabulary	Type count (r = 0.73) Mean word length (r = 0.16) Word frequency measure (r = 0.63)
Narrative	Linking words (*however, moreover, nevertheless*) per token (r = 0.02)
Complexity	Mean sentence length (r = −0.02) Semicolons per token (r = 0.06) Prepositional phrases per token (r = 0.25) Depth of longest branch in dependency tree (r = 0.37)

For the frequency measure, I built a frequency list from the American National Corpus (anc.org)…



…verified that it conformed to Zipf's Law, that word type frequencies are power law distributed…



…and calculated the sum of the token ranks.
(This is an arbitrary measure I made up.)

"Because I could not stop for Death, he kindly stopped for me." → **9,164**

"The boy stood on the burning deck whence all but he had fled." → **23,815**

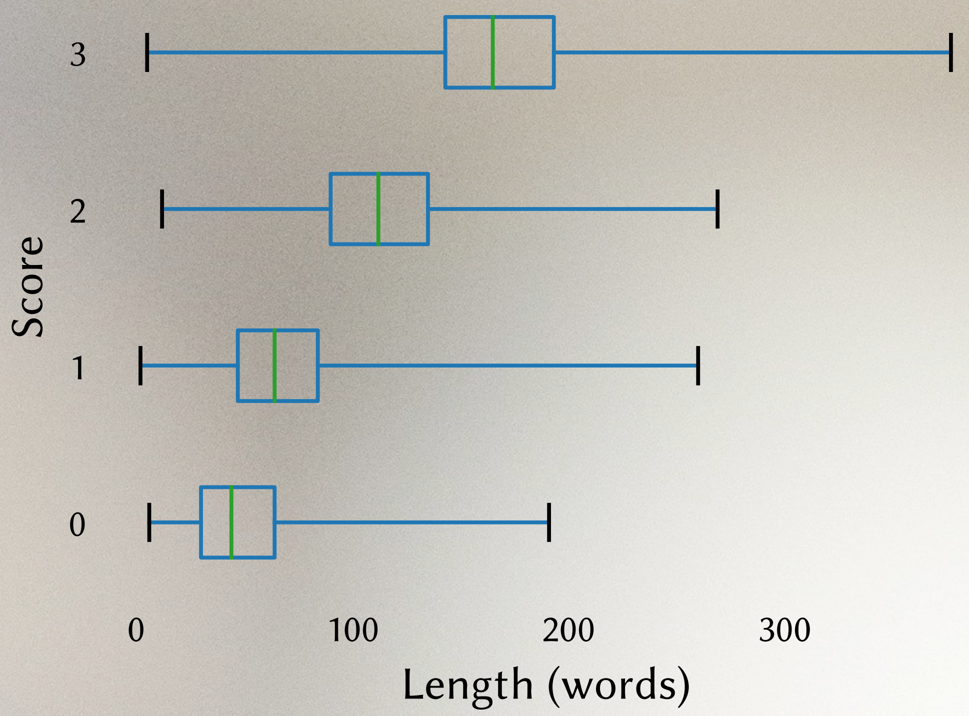
"I was the shadow of the waxwing slain by the false azure in the windowpane." → **86,636**

"'Twas brillig, and the slithy toves did gyre and gimble in the wabe." → **146,103**

The language parsing was done with spaCy (spacy.io), which is fast (written in C) and dead simple to use.

```
import spacy

# Load English data
nlp = spacy.load("en")

[(t, t.lemma_, t.pos_) for t in
 nlp("The cat sat on the mat")]

# Output:
[(The, 'the', 'DET'),
 (cat, 'cat', 'NOUN'),
 (sat, 'sit', 'VERB'),
 (on,  'on',  'ADP'),
 (the, 'the', 'DET'),
 (mat, 'mat', 'NOUN')]
```

Length correlates with score (r = 0.72): all else equal, 82 more words yield a unit increase in score ($p \ll 0.01$).



…but some top-scoring essays are ludicrously short:

The features of the setting affect the cyclist in many ways. It made him tired thirsty and he was near exaustion.

If it's a good day the cyclist will want to rid. If it is a bad day then they will not want to but probably will.

He is going on a journey on his bike. He has to go down hills and deal with the factt that has has no water. That's basically what the setting is about.

Reserved need to check keenly

…but some top-scoring essays are ludicrously short:

The features of the setting affect the cyclist in many ways. It made him tired thirsty and he was near exaustion.

If it's a good day the cyclist will want to rid. If it is a bad day then they will not want to but probably will.

He is going on a journey on his bike. He has to go down hills and deal with the factt that has has no water. That's basically what the setting is about.

Reserved need to check keenly

Without an objective way of pruning these, I decided to leave them in the dataset.

k-NN with 4 centroids revealed next to nothing.

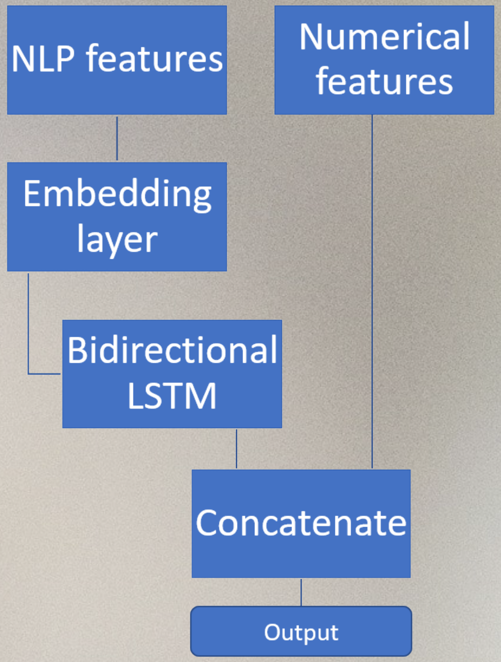
Conventional models yielded modest test scores:

Baseline: 36%
Linear regression: **57%**
LASSO regression: **57%** $\alpha = 0.01$
Gaussian naïve Bayes: **53%**
Support vector machine: **64%** $\alpha = 1$
AdaBoost: **58%**
ExtraTrees: **64%** 1,000 estimators, max depth = 300

Conventional models yielded modest test scores:

Baseline: 36%
Linear regression: **57%**
LASSO regression: **57%** $\alpha = 0.01$
Gaussian naïve Bayes: **53%**
Support vector machine: **64%** $\alpha = 1$
AdaBoost: **58%**
ExtraTrees: **64%** 1,000 estimators, max depth = 300
RNN?

An RNN can input both a text series (“NLP features”) and metadata about the text (“Numerical features”).

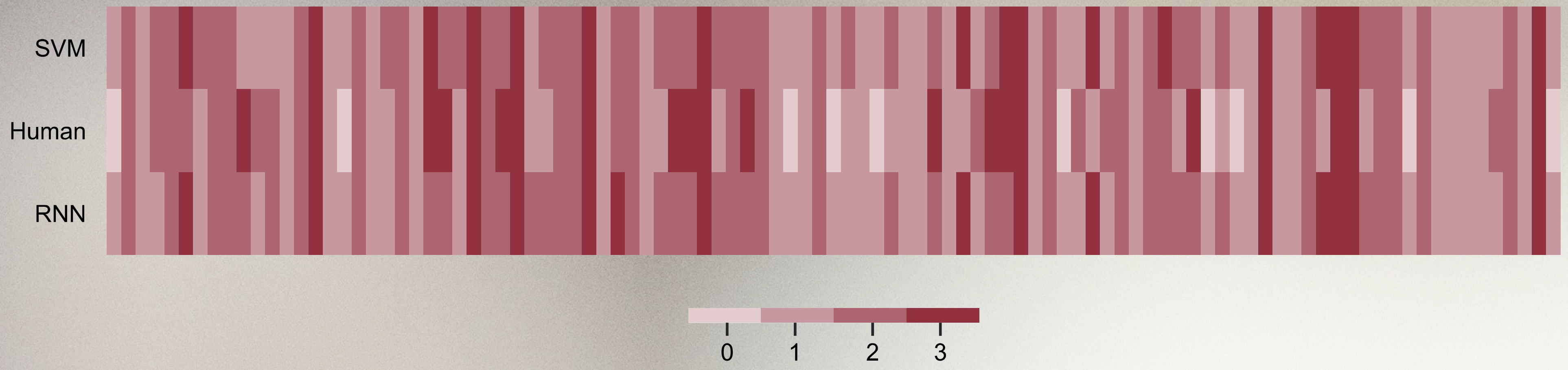


```
# Define inputs
vector_input = Input(shape=(1000,)) # Text vectors, in series of length 1,000
meta_input = Input(shape=(5,)) # PCA-transformed metadata (types, tokens, etc.)

# Embedding layer turns lists of word indices into dense vectors
rnn = Embedding(
        input_dim = len(vocab),
        output_dim = 96,
        input_length = 1000
    )(vector_input)

# GRU layers for RNN
rnn = Bidirectional(GRU(256, return_sequences=True, kernel_regularizer=l2(0.01)))(rnn)
rnn = Bidirectional(GRU(256, return_sequences=False, kernel_regularizer=l2(0.01)))(rnn)

# Incorporate metadata
rnn = Concatenate()([rnn, meta_input])

# Define hidden and output layers
rnn = Dense(128, activation="relu", kernel_regularizer=l2(0.01))(rnn)
rnn = Dense(128, activation="relu", kernel_regularizer=l2(0.01))(rnn)
rnn = Dense(4, activation="softmax")(rnn)

# Define model
model = Model(inputs=[vector_input, meta_input], outputs=[rnn])
```

But it didn’t do much better than the others:

Baseline: 36%
Linear regression: **57%**
LASSO regression: **57%** $\alpha = 0.01$
Gaussian naïve Bayes: **53%**
Support vector machine: **64%** $\alpha = 1$
AdaBoost: **58%**
ExtraTrees: **64%** 1,000 estimators, maximum depth = 300
RNN: **62%**

The SVM and RNN predicted similar scores, probably since they were given the same metadata.



The trained RNN model is packaged into an independent utility: EssayScorer.py (~150 lines).

```
2 alex pequod:~/C/G/P/C/EssayScorer master$ ./EssayScorer.py
Type your essay:
He is going on a journey on his bike. He has to go down hills and deal with the factt
that has has no water. That's basically what the setting is about.
Loading NLP data...
Loading model...
Preprocessing essay...
Running model...
Score: 1
```

Final tally:

- ✓ Completion
- ? Structure
- ✗ Relevance
- ✗ Grammar/usage/mechanics

Thank you!

Appendix: Python libraries

```
@show_progress
def my_fun(i, item):
    ... do_stuff_to(item) ...

my_fun(my_list)



Parsing item 1,969 of 12,976 [=====>                         ] 15%, 38:45 remains...
```