In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:
```
import statsmodels.formula.api as smf
from sklearn.metrics import r2_score
```

In [4]:
```
from sklearn.linear_model import LinearRegression
```

In [5]:
```
from IPython.display import HTML
```

In [6]:
```
import statsmodels.api as sm
```

```
/Users/kcarnold/anaconda3/envs/py36/lib/python3.6/site-packages/statsmodels/comp
at/pandas.py:56: FutureWarning: The pandas.core.datetools module is deprecated a
nd will be removed in a future version. Please use the pandas.tseries module ins
tead.
  from pandas.core import datetools
```

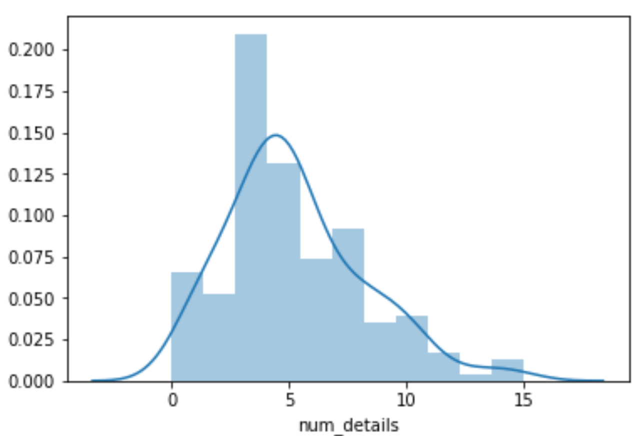
In [7]:
```
from textrec.paths import paths
```

In [8]:
```
dataset = pd.read_csv(paths.data / 'num_details_training_set.csv')
```

In [9]:
```
sns.distplot(dataset.num_details)
```

```
/Users/kcarnold/anaconda3/envs/py36/lib/python3.6/site-packages/scipy/stats/stat
s.py:1706: FutureWarning: Using a non-tuple sequence for multidimensional indexi
ng is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future thi
s will be interpreted as an array index, `arr[np.array(seq)]`, which will result
either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c1afe02e8>
```



Preprocessing

Strip off punctuation; it just throws off token counts and probs. (We get a few percent boost in r^2 because of this.)

In [10]:
```
dataset['text'] = dataset.text.str.strip().str.rstrip('.')
```

In [11]:
```
def strip_uninformative(text):
    text = text.strip()
    for beginning in ['there is', 'there are', 'a view of', 'a photo of', 'a photo
shows']:
        beginning = beginning + ' '
        if text.startswith(beginning):
            text = text[len(beginning):]
            return strip_uninformative(text)
    return text
```

In [12]:
```
strip_uninformative('there is a view of a red thing there')
```

Out[12]:
```
'a red thing there'
```

In [13]:
```
dataset['text'] = dataset.text.apply(strip_uninformative)
```

Word Frequencies

In [14]:
```
import wordfreq
```

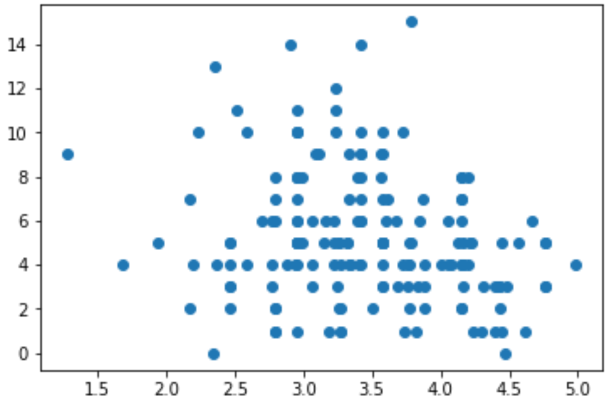
In [15]:
```
dataset['num_words'] = [
    len(wordfreq.tokenize(text, 'en'))
    for text in dataset.text]
```

In [16]:
```
dataset['min_freq'] = [
    np.min([wordfreq.zipf_frequency(tok, 'en') for tok in wordfreq.tokenize(text,
'en')])
    for text in dataset.text]
```

In [17]:
```
dataset['mean_freq'] = [
    np.mean([wordfreq.zipf_frequency(tok, 'en') for tok in wordfreq.tokenize(text,
'en')])
    for text in dataset.text]
```

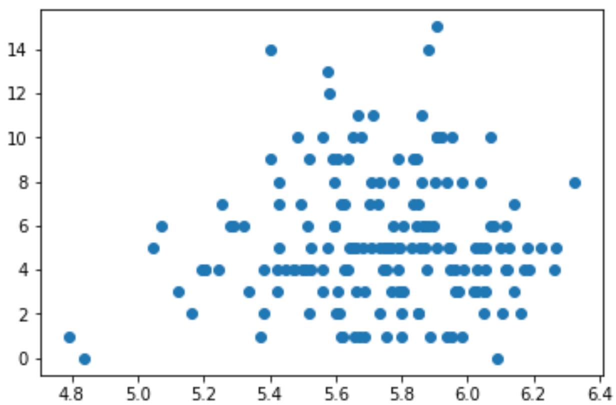
In [18]:
```
plt.scatter(dataset.min_freq, dataset.num_details)
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x1c1de300f0>
```



In [19]:
```
plt.scatter(dataset.mean_freq, dataset.num_details)
```

Out[19]:
```
<matplotlib.collections.PathCollection at 0x1c1deae898>
```



Perplexity

The perplexity of a language model is a rough proxy for the amout of information that a text contains. The more details included, the more uncertainty the LM has; and redundant text doesn't get counted. It's not quite right for a few reasons:

- Typos, grammar errors, etc. also increase perplexity
- Unusual wording of the same concepts increases perplexity
- Using a word that's *more* common than expected increases perplexity.

But we'll try it anyway.

In [20]:
```
from textrec import automated_analyses
from textrec import onmt_model_2
```

```
/Users/kcarnold/anaconda3/envs/py36/lib/python3.6/site-packages/h5py/__init__.py
:36: FutureWarning: Conversion of the second argument of issubdtype from `float`
to `np.floating` is deprecated. In future, it will be treated as `np.float64 ==
np.dtype(float).type`.
  from ._conv import register_converters as _register_converters

Loading ONMT models...
coco_lm_adam_acc_46.00_ppl_16.32_e10_nooptim.pt
Loading model parameters.
coco_cap_adam_acc_48.73_ppl_12.56_e10_nooptim.pt
Loading model parameters.
Ready.
Loading SpaCy...done
```

In [21]:
```
automated_analyses.eval_logprobs_unconditional(dataset.text.iloc[0])
```

Out[21]:
```
3.3479643
```

In [22]:
```
example_text = dataset.text.iloc[0]
example_text
```

Out[22]:
```
'families stand around by the water flying kites on a sunny day'
```

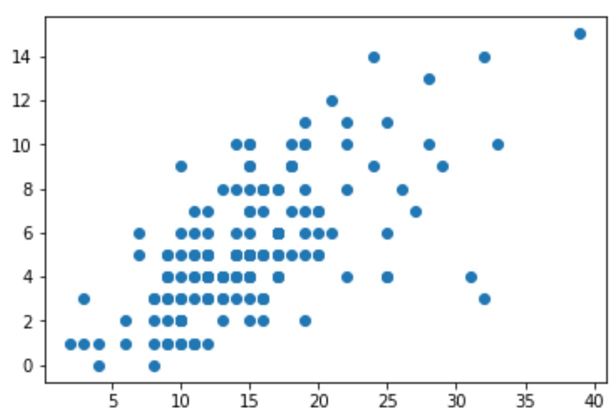
In [23]:
```
tokens = onmt_model_2.tokenize(example_text)
logprobs = onmt_model_2.models['coco_lm'].eval_logprobs('.', tokens, use_eos=True)
logprobs
```

Out[23]:
```
array([1.0859766e+01, 4.1381788e+00, 2.3187706e+00, 7.0211720e+00,
       1.5096430e+00, 2.1149969e+00, 5.2354274e+00, 3.4142053e-01,
       5.1307883e+00, 1.5549884e+00, 1.9299134e+00, 3.4333759e-03,
       1.3650393e+00], dtype=float32)
```

In [24]:
```
dataset['num_tokens'] = dataset.text.apply(lambda text: len(onmt_model_2.tokenize(
text)))
dataset['mean_logprob_uncond'] = dataset.text.apply(lambda text: automated_analyse
s.eval_logprobs_unconditional(text))
dataset['total_logprob_uncond'] = dataset.mean_logprob_uncond * (dataset.num_token
s + 1)
```

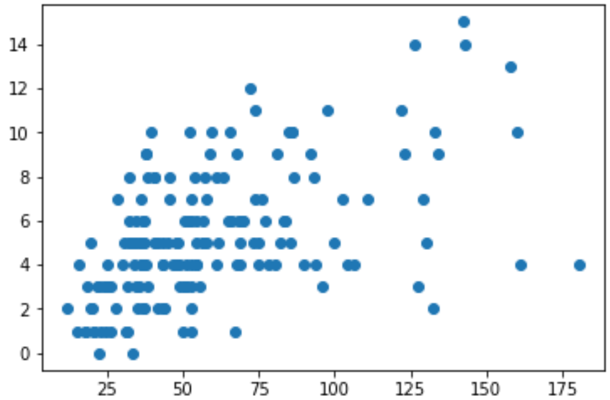
In [27]:
```
plt.scatter(dataset.num_tokens, dataset.num_details)
```

Out[27]:
```
<matplotlib.collections.PathCollection at 0x1c432d8d30>
```



In [28]:
```
plt.scatter(dataset.total_logprob_uncond, dataset.num_details)
```

Out[28]:
```
<matplotlib.collections.PathCollection at 0x1c34f0da90>
```



Models

In [29]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168 entries, 0 to 167
Data columns (total 9 columns):
image_id                168 non-null int64
text                    168 non-null object
num_details             168 non-null int64
num_words               168 non-null int64
min_freq                168 non-null float64
mean_freq               168 non-null float64
num_tokens              168 non-null int64
mean_logprob_uncond     168 non-null float64
total_logprob_uncond    168 non-null float64
dtypes: float64(4), int64(4), object(1)
memory usage: 11.9+ KB
```

In [50]:
```
dataset[dataset.num_words != dataset.num_tokens][['text', 'num_words', 'num_tokens
']]
```

Out[50]:

	text	num_words	num_tokens
14	a man in a red shirt with two children are on a beach holding a multi-colored kite while other people fly kites in the background	26	25
19	a man in a red shirt is helping his children fly a large rainbow-colored kite	16	15
22	the image shows a railroad track with a train on it further out in the distance. multiple white buildings hug the side if the track, with some woo...	30	31
35	a train passing a few small buildings, perhaps the station	10	11
41	a landscape of a train stop with an old-looking brownish train and a few brightly colored buildings to one side	21	20
43	a black-and-white picture of a young couple cutting the wedding cake with the help of a young photographer at the wedding event	24	22
45	a woman is standing next to a couple in front of a cake with a knife in it and holding the other woman's hands	24	25
56	a husband, bride and female all stand in front of a table holding a knife cutting a cake	18	19
80	a man-woman gracefully riding a wave using a surfboard	10	9
81	a surfer is riding a wave the water looks so refreshing it's a beautiful day	15	16
86	a double-decker bus drives through a busy city street in london	12	11
89	a busy city street with cars, a large red bus and pedestrians going about their day	16	17
91	the photo shows a downtown scene of a city. there are old buildings everywhere, and a red bus is prominent in the middle of the road. many people ...	32	33
97	a busy city street with cars and people along the streets with high-rise buildings on both sides	18	17
99	a red double-decker bus driving down a street next to tall buildings and a cloudy sky in london	19	18
101	a red double-decker bus passes a group of people to its left while a black car looks to pass	20	19
111	a curious cat sits perched upon a table, next to a glass of wine	14	15
118	a brownish-orange cat with yellow eyes is look to his left past a glass of red wine	18	17
128	sliding glass, frosted, shower doors with a tan towel hanging on the handle and a white toilet with a blue floor rug	22	24
141	someone is using a shower but it's hard to see due to the opaque glass	15	16
147	a toilet paper sits on top of a toilet next to the sink, in a plain bathroom	17	18
153	a toilet has a roll of toilet paper on it, and there is a sink that matches it to the right	21	22
155	a sink, mirror and toilet, all in white with a roll of toilet paper on the toilet	17	19

In [30]:
```
dataset.mean_freq.describe()
```

Out[30]:
```
count    168.000000
mean       5.738019
std        0.283893
min        4.790000
25%        5.579911
50%        5.764444
75%        5.942990
max        6.323333
Name: mean_freq, dtype: float64
```

Let's try including the interaction of mean_freq and tokens. That's sorta like the total word frequency.. if we invert frequency to make `rarity` , then it's total rarity, or something proportional to unigram perplexity.

In [31]:
```
dataset['mean_rarity'] = (7 - dataset.mean_freq) / 7
dataset['max_rarity'] = (7 - dataset.min_freq) / 7
dataset['total_rarity'] = dataset['mean_rarity'] * dataset['num_words']
```

In [32]:
```
formulas = '''
C(image_id) + min_freq + mean_freq
C(image_id) + total_rarity
C(image_id) + num_tokens + total_rarity
C(image_id) + num_tokens + total_rarity + total_logprob_uncond + mean_logprob_unco
nd
C(image_id) + num_tokens + mean_rarity + max_rarity + total_rarity + total_logprob
_uncond + mean_logprob_uncond + max_rarity*num_tokens
C(image_id) + num_tokens + mean_rarity + max_rarity + total_rarity + max_rarity*nu
m_tokens
'''.split('\n')

models = {}
for formula in formulas:
    formula = formula.strip()
    if not formula:
        continue
    formula_full = 'num_details ~ ' + formula
    models[formula] = model = smf.ols(formula_full, dataset).fit()
    display(HTML(f'<h1>r^2={model.rsquared:.3f}: {formula}</h1>'))
    display(model.summary())
```

r^2=0.385: C(image_id) + min_freq + mean_freq

OLS Regression Results

Dep. Variable:	num_details	**R-squared:**	0.385
Model:	OLS	**Adj. R-squared:**	0.350
Method:	Least Squares	**F-statistic:**	10.98
Date:	Wed, 10 Oct 2018	**Prob (F-statistic):**	3.40e-13
Time:	09:49:30	**Log-Likelihood:**	-378.74
No. Observations:	168	**AIC:**	777.5
Df Residuals:	158	**BIC:**	808.7
Df Model:	9		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	3.2883	3.939	0.835	0.405	-4.492	11.068
C(image_id)[T.223777]	-2.3113	0.802	-2.882	0.005	-3.896	-0.727
C(image_id)[T.227326]	-2.1374	0.752	-2.841	0.005	-3.624	-0.651
C(image_id)[T.240275]	-4.3229	0.736	-5.877	0.000	-5.776	-2.870
C(image_id)[T.247576]	1.4757	0.754	1.956	0.052	-0.014	2.966
C(image_id)[T.275449]	-1.2033	0.749	-1.607	0.110	-2.682	0.276
C(image_id)[T.396295]	-1.1903	0.757	-1.572	0.118	-2.686	0.305
C(image_id)[T.431140]	0.1770	0.772	0.229	0.819	-1.349	1.703
min_freq	-1.5700	0.342	-4.595	0.000	-2.245	-0.895
mean_freq	1.4900	0.740	2.015	0.046	0.029	2.951

Omnibus:	17.294	**Durbin-Watson:**	2.051
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	23.230
Skew:	0.628	**Prob(JB):**	9.03e-06
Kurtosis:	4.320	**Cond. No.**	148.

r^2=0.713: C(image_id) + total_rarity

OLS Regression Results

Dep. Variable:	num_details	**R-squared:**	0.713
Model:	OLS	**Adj. R-squared:**	0.699
Method:	Least Squares	**F-statistic:**	49.49
Date:	Wed, 10 Oct 2018	**Prob (F-statistic):**	2.32e-39
Time:	09:49:30	**Log-Likelihood:**	-314.55
No. Observations:	168	**AIC:**	647.1
Df Residuals:	159	**BIC:**	675.2
Df Model:	8		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	2.2352	0.473	4.722	0.000	1.300	3.170
C(image_id)[T.223777]	-3.6844	0.499	-7.381	0.000	-4.670	-2.698
C(image_id)[T.227326]	-3.6428	0.503	-7.243	0.000	-4.636	-2.649
C(image_id)[T.240275]	-3.6986	0.500	-7.399	0.000	-4.686	-2.711
C(image_id)[T.247576]	-0.1979	0.503	-0.393	0.695	-1.192	0.796
C(image_id)[T.275449]	-1.7292	0.499	-3.464	0.001	-2.715	-0.743
C(image_id)[T.396295]	-1.9313	0.499	-3.868	0.000	-2.917	-0.945
C(image_id)[T.431140]	-0.9063	0.500	-1.813	0.072	-1.893	0.081
total_rarity	1.9068	0.126	15.109	0.000	1.658	2.156

Omnibus:	1.608	**Durbin-Watson:**	1.877
Prob(Omnibus):	0.447	**Jarque-Bera (JB):**	1.227
Skew:	-0.183	**Prob(JB):**	0.541
Kurtosis:	3.204	**Cond. No.**	25.2

r^2=0.736: C(image_id) + num_tokens + total_rarity

OLS Regression Results

Dep. Variable:	num_details	**R-squared:**	0.736
Model:	OLS	**Adj. R-squared:**	0.721
Method:	Least Squares	**F-statistic:**	48.90
Date:	Wed, 10 Oct 2018	**Prob (F-statistic):**	3.07e-41
Time:	09:49:30	**Log-Likelihood:**	-307.73
No. Observations:	168	**AIC:**	635.5
Df Residuals:	158	**BIC:**	666.7
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.0451	0.459	4.457	0.000	1.139	2.951
C(image_id)[T.223777]	-3.9423	0.486	-8.112	0.000	-4.902	-2.982
C(image_id)[T.227326]	-4.0243	0.496	-8.120	0.000	-5.003	-3.045
C(image_id)[T.240275]	-3.6838	0.482	-7.650	0.000	-4.635	-2.733
C(image_id)[T.247576]	-0.4408	0.489	-0.901	0.369	-1.407	0.526
C(image_id)[T.275449]	-1.8718	0.482	-3.880	0.000	-2.825	-0.919
C(image_id)[T.396295]	-1.9181	0.481	-3.988	0.000	-2.868	-0.968
C(image_id)[T.431140]	-1.3426	0.496	-2.707	0.008	-2.322	-0.363
num_tokens	0.1481	0.041	3.656	0.000	0.068	0.228
total_rarity	1.1980	0.229	5.236	0.000	0.746	1.650

Omnibus:	5.045	**Durbin-Watson:**	1.760
Prob(Omnibus):	0.080	**Jarque-Bera (JB):**	5.034
Skew:	-0.281	**Prob(JB):**	0.0807
Kurtosis:	3.635	**Cond. No.**	140.

r^2=0.743: C(image_id) + num_tokens + total_rarity + total_logprob_uncond + mean_logprob_uncond

OLS Regression Results

Dep. Variable:	num_details	**R-squared:**	0.743
Model:	OLS	**Adj. R-squared:**	0.725
Method:	Least Squares	**F-statistic:**	41.00
Date:	Wed, 10 Oct 2018	**Prob (F-statistic):**	1.84e-40
Time:	09:49:30	**Log-Likelihood:**	-305.41
No. Observations:	168	**AIC:**	634.8
Df Residuals:	156	**BIC:**	672.3
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.7258	1.160	0.625	0.533	-1.566	3.018
C(image_id)[T.223777]	-3.7699	0.490	-7.699	0.000	-4.737	-2.803
C(image_id)[T.227326]	-4.0356	0.493	-8.182	0.000	-5.010	-3.061
C(image_id)[T.240275]	-3.6715	0.478	-7.674	0.000	-4.617	-2.726
C(image_id)[T.247576]	-0.4646	0.486	-0.956	0.340	-1.424	0.495
C(image_id)[T.275449]	-1.8630	0.480	-3.878	0.000	-2.812	-0.914
C(image_id)[T.396295]	-1.7238	0.492	-3.502	0.001	-2.696	-0.751
C(image_id)[T.431140]	-1.3638	0.493	-2.768	0.006	-2.337	-0.391
num_tokens	0.2491	0.081	3.091	0.002	0.090	0.408
total_rarity	1.3737	0.246	5.574	0.000	0.887	1.861
total_logprob_uncond	-0.0292	0.018	-1.637	0.104	-0.064	0.006
mean_logprob_uncond	0.2810	0.277	1.014	0.312	-0.266	0.828

Omnibus:	3.726	**Durbin-Watson:**	1.837
Prob(Omnibus):	0.155	**Jarque-Bera (JB):**	3.774
Skew:	-0.180	**Prob(JB):**	0.152
Kurtosis:	3.640	**Cond. No.**	731.

r^2=0.745: C(image_id) + num_tokens + mean_rarity + max_rarity + total_rarity + total_logprob_uncond + mean_logprob_uncond + max_rarity*num_tokens

OLS Regression Results

Dep. Variable:	num_details	**R-squared:**	0.745
Model:	OLS	**Adj. R-squared:**	0.722
Method:	Least Squares	**F-statistic:**	31.92
Date:	Wed, 10 Oct 2018	**Prob (F-statistic):**	2.57e-38
Time:	09:49:30	**Log-Likelihood:**	-304.77
No. Observations:	168	**AIC:**	639.5
Df Residuals:	153	**BIC:**	686.4
Df Model:	14		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1.8745	2.096	0.894	0.372	-2.266	6.015
C(image_id)[T.223777]	-3.7092	0.551	-6.738	0.000	-4.797	-2.622
C(image_id)[T.227326]	-4.0110	0.514	-7.803	0.000	-5.026	-2.996
C(image_id)[T.240275]	-3.6637	0.485	-7.551	0.000	-4.622	-2.705
C(image_id)[T.247576]	-0.4367	0.513	-0.852	0.396	-1.450	0.576
C(image_id)[T.275449]	-1.8771	0.496	-3.784	0.000	-2.857	-0.897
C(image_id)[T.396295]	-1.7201	0.520	-3.308	0.001	-2.748	-0.693
C(image_id)[T.431140]	-1.3342	0.516	-2.584	0.011	-2.354	-0.314
num_tokens	0.1609	0.126	1.274	0.204	-0.089	0.410
mean_rarity	-3.8772	8.361	-0.464	0.643	-20.394	12.640
max_rarity	-1.5791	3.463	-0.456	0.649	-8.421	5.263
total_rarity	1.5233	0.545	2.796	0.006	0.447	2.600
total_logprob_uncond	-0.0360	0.020	-1.797	0.074	-0.076	0.004
mean_logprob_uncond	0.3996	0.332	1.205	0.230	-0.256	1.055
max_rarity:num_tokens	0.1615	0.204	0.792	0.430	-0.241	0.564

Omnibus:	4.007	**Durbin-Watson:**	1.867
Prob(Omnibus):	0.135	**Jarque-Bera (JB):**	4.201
Skew:	-0.183	**Prob(JB):**	0.122
Kurtosis:	3.683	**Cond. No.**	4.92e+03

r^2=0.737: C(image_id) + num_tokens + mean_rarity + max_rarity + total_rarity + max_rarity*num_tokens

OLS Regression Results

Dep. Variable:	num_details	**R-squared:**	0.737
Model:	OLS	**Adj. R-squared:**	0.716
Method:	Least Squares	**F-statistic:**	36.13
Date:	Wed, 10 Oct 2018	**Prob (F-statistic):**	7.70e-39
Time:	09:49:30	**Log-Likelihood:**	-307.47
No. Observations:	168	**AIC:**	640.9
Df Residuals:	155	**BIC:**	681.5
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.1419	2.030	1.548	0.124	-0.868	7.152
C(image_id)[T.223777]	-3.9445	0.545	-7.241	0.000	-5.021	-2.868
C(image_id)[T.227326]	-4.0215	0.519	-7.756	0.000	-5.046	-2.997
C(image_id)[T.240275]	-3.6782	0.489	-7.529	0.000	-4.643	-2.713
C(image_id)[T.247576]	-0.4441	0.518	-0.858	0.392	-1.467	0.578
C(image_id)[T.275449]	-1.9017	0.501	-3.799	0.000	-2.891	-0.913
C(image_id)[T.396295]	-1.9284	0.506	-3.813	0.000	-2.928	-0.929
C(image_id)[T.431140]	-1.3438	0.521	-2.578	0.011	-2.373	-0.314
num_tokens	0.0734	0.121	0.608	0.544	-0.165	0.312
mean_rarity	-1.7638	6.952	-0.254	0.800	-15.498	11.970
max_rarity	-1.4324	3.492	-0.410	0.682	-8.331	5.466
total_rarity	1.2691	0.492	2.581	0.011	0.298	2.241
max_rarity:num_tokens	0.1159	0.205	0.567	0.572	-0.288	0.520

Omnibus:	4.893	**Durbin-Watson:**	1.774
Prob(Omnibus):	0.087	**Jarque-Bera (JB):**	4.818
Skew:	-0.279	**Prob(JB):**	0.0899
Kurtosis:	3.614	**Cond. No.**	1.08e+03

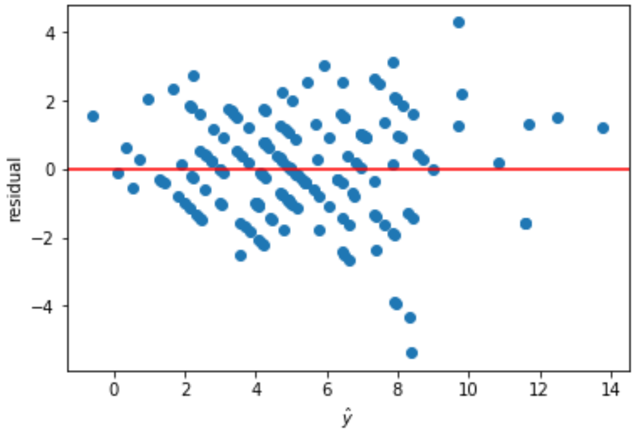
Summary: num_details increases by 1.4 for each additional token of rarity.

Let's look at resids.

In [33]:
```
model = models['C(image_id) + num_tokens + total_rarity']
```

In [34]:
```
predicted = model.predict(dataset)
```

In [35]:
```
plt.scatter(predicted, model.resid)
plt.axhline(0, color='r')
plt.xlabel('$\hat{y}$')
plt.ylabel('residual');
```



Ok, let's have a look at captions for which length and frequency don't predict num_details well.

In [36]:
```
dsr = dataset.copy()
```

In [37]:
```
dsr['resid'] = model.resid
dsr['resid_mag'] = model.resid.abs()
dsr['predicted'] = predicted
dsr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168 entries, 0 to 167
Data columns (total 15 columns):
image_id                168 non-null int64
text                    168 non-null object
num_details             168 non-null int64
num_words               168 non-null int64
min_freq                168 non-null float64
mean_freq               168 non-null float64
num_tokens              168 non-null int64
mean_logprob_uncond     168 non-null float64
total_logprob_uncond    168 non-null float64
mean_rarity             168 non-null float64
max_rarity              168 non-null float64
total_rarity            168 non-null float64
resid                   168 non-null float64
resid_mag               168 non-null float64
predicted               168 non-null float64
dtypes: float64(10), int64(4), object(1)
memory usage: 19.8+ KB
```

In [38]:
```
pd.set_option('display.max_colwidth', 150)
```

In [39]:
```
print("over-predicted:")
dsr[dsr.predicted.between(4,8)]['image_id text num_tokens resid predicted total_lo
gprob_uncond num_details'.split()].sort_values('resid').iloc[:5]
```

```
over-predicted:
```

Out[39]:

	image_id	text	num_tokens	resid	predicted	total_logprob_uncond	num_details
3	200451	several multicolored kites with streamers are seen soaring above the heads of people	13	-3.965627	7.965627	54.609120	4
22	223777	the image shows a railroad track with a train on it further out in the distance. multiple white buildings hug the side if the track, with some woo...	31	-3.901685	7.901685	180.252899	4
9	200451	one kite flying over four other kites on a blue sky	11	-2.645860	6.645860	43.533666	4
103	247576	a double decker bus traveling down the middle of the street in the city streets	15	-2.493046	6.493046	36.642353	4
164	431140	toilet paper roll is on top of the toilet in a mellow yellow painted bathroom	15	-2.422958	6.422958	75.009583	4

In [40]:
```
print("Under-predicted")
dsr[dsr.predicted.between(4,8)]['image_id text num_tokens resid predicted total_lo
gprob_uncond num_details'.split()].sort_values('resid').iloc[-5:]
```

```
Under-predicted
```

Out[40]:

	image_id	text	num_tokens	resid	predicted	total_logprob_uncond	num_details
120	275449	a half full glass of red wine on a table in front of a calico cat	16	2.541694	5.458306	38.410653	8
145	396295	a tan towel is hanging from a chrome handle on a textured glass shower door	15	2.543200	6.456800	81.209793	9
2	200451	a man and his two children are flying multicolored kites on a sandy beach	14	2.646132	7.353868	39.750552	10
7	200451	a man flies a butterfly kite with his two daughters	10	3.061945	5.938055	37.889245	9
151	431140	a bathroom with a white sink and white toilet. a roll of unwrapped toilet paper sits on the bowl	19	3.144114	7.855886	97.487974	11

I notice:

- We can generally do surprisingly well on this task using total rarity. We can explain about 74% of the variance in details.
- Some of the over-predicts actually have more details than I gave them credit for. Some of the under-predicts are less detailed.
- Some of the over-predicted just have extra words ("there is" one kite; "a view of" a bathroom, "in the city streets"); I went back and stripped them off and the above reflects that. (we get a boost of about 0.01 R^2.)

Since this model has image only as a slope (should be random but alas I'm lazy), we can still get relative details measures.

Aside: random-effects model.

In [41]:
```
md = smf.mixedlm("num_details ~ mean_rarity + num_tokens + total_rarity", dataset,
groups=dataset["image_id"])

md.fit().summary()
```

Out[41]:

Model:	MixedLM	Dependent Variable:	num_details
No. Observations:	168	Method:	REML
No. Groups:	8	Scale:	2.4408
Min. group size:	21	Likelihood:	-323.9069
Max. group size:	21	Converged:	Yes
Mean group size:	21.0		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
Intercept	0.371	1.431	0.259	0.795	-2.434	3.177
mean_rarity	-2.535	6.720	-0.377	0.706	-15.707	10.636
num_tokens	0.118	0.085	1.392	0.164	-0.048	0.285
total_rarity	1.360	0.465	2.924	0.003	0.449	2.272
groups RE	2.372	0.870				

In [42]:
```
md = smf.mixedlm("num_details ~ total_rarity", dataset, groups=dataset["image_id"]
)
mdf = md.fit()
mdf.summary()
```

Out[42]:

Model:	MixedLM	Dependent Variable:	num_details
No. Observations:	168	Method:	REML
No. Groups:	8	Scale:	2.6164
Min. group size:	21	Likelihood:	-330.8710
Max. group size:	21	Converged:	Yes
Mean group size:	21.0		

	Coef.	**Std.Err.**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	0.251	0.641	0.392	0.695	-1.006	1.508
total_rarity	1.911	0.126	15.161	0.000	1.664	2.158
groups RE	2.295	0.817				

In [43]:
```
r2_score(dataset['num_details'], mdf.predict(dataset))
```

Out[43]:
```
0.4685851105546911
```

I don't understand why that R^2 score is much smaller than the fixed-effects version. Probably we have different parameters.

In [44]:
```
model.params
```

Out[44]:
```
Intercept                    2.045098
C(image_id)[T.223777]       -3.942306
C(image_id)[T.227326]       -4.024281
C(image_id)[T.240275]       -3.683757
C(image_id)[T.247576]       -0.440750
C(image_id)[T.275449]       -1.871804
C(image_id)[T.396295]       -1.918077
C(image_id)[T.431140]       -1.342626
num_tokens                   0.148146
total_rarity                 1.198048
dtype: float64
```

In [45]:
```
mdf.params
```

Out[45]:
```
Intercept       0.251184
total_rarity    1.910701
groups RE       0.877341
dtype: float64
```

In [46]:
```
re_params = pd.Series({k: v.iloc[0] for k, v in mdf.random_effects.items()})
re_params
```

Out[46]:
```
200451    1.872634
223777   -1.622234
227326   -1.584410
240275   -1.634757
247576    1.683106
275449    0.232425
396295    0.041020
431140    1.012215
dtype: float64
```

In [47]:
```
fixed_params = pd.Series({int(k[14:-1]): v for k, v in model.params.items() if k.s
tartswith('C')})
fixed_params
```

Out[47]:
```
223777   -3.942306
227326   -4.024281
240275   -3.683757
247576   -0.440750
275449   -1.871804
396295   -1.918077
431140   -1.342626
dtype: float64
```

In [48]:
```
d = pd.DataFrame(dict(fixed=fixed_params + model.params['Intercept'], random=re_pa
rams + mdf.params['Intercept']))
d['diff'] = d['fixed'] - d['random']
d['absdiff'] = d['fixed'].abs() - d['random'].abs()
d.mean(axis=0)
```

Out[48]:
```
fixed     -0.415417
random     0.251184
diff      -0.399082
absdiff    0.008690
dtype: float64
```

The random-effects estimates are generally smaller.