In [5]:
```
from fastai import *          # Quick access to most common functionality
from fastai.tabular import *  # Quick access to tabular functionality
from bs4 import BeautifulSoup
```

# Tabular example

Tabular data should be in a Pandas `DataFrame`.

In [2]:
```
def read_json_line(line=None):
    result = None
    try:
        result = json.loads(line)
    except Exception as e:
        # Find the offending character index:
        idx_to_replace = int(str(e).split(' ')[-1].replace(')',''))
        # Remove the offending character:
        new_line = list(line)
        new_line[idx_to_replace] = ' '
        new_line = ''.join(new_line)
        return read_json_line(line=new_line)
    return result

from html.parser import HTMLParser

class MLStripper(HTMLParser):
    def __init__(self):
        self.reset()
        self.strict = False
        self.convert_charrefs= True
        self.fed = []
    def handle_data(self, d):
        self.fed.append(d)
    def get_data(self):
        return ''.join(self.fed)

def strip_tags(html):
    s = MLStripper()
    s.feed(html)
    return s.get_data()

def extract_features(path_to_data):

    content_list = []
    published_list = []
    title_list = []
    author_list = []
    domain_list = []
    tags_list = []
    url_list = []

    with open(path_to_data, encoding='utf-8') as inp_json_file:
        for line in inp_json_file:
            json_data = read_json_line(line)
            content = json_data['content'].replace('\n', ' ').replace('\r', ' '
)
            content_no_html_tags = strip_tags(content)
            content_list.append(content_no_html_tags)
            published = json_data['published']['$date']
            published_list.append(published)
            title = json_data['meta_tags']['title'].split('\u2013')[0].strip()
#'Medium Terms of Service – Medium Policy – Medium'
            title_list.append(title)
            author = json_data['meta_tags']['author'].strip()
            author_list.append(author)
            domain = json_data['domain']
            domain_list.append(domain)
            url = json_data['url']
            url_list.append(url)

            tags_str = []
            soup = BeautifulSoup(content, 'lxml')
```

In [3]:
```
PATH = "../data/"
```

In [6]:
```
content_list, published_list, title_list, author_list, domain_list, tags_list,
url_list = extract_features(os.path.join(PATH, 'medium/train.json'))
train = pd.DataFrame()
train['content'] = content_list
train['published'] = pd.to_datetime(published_list, format='%Y-%m-%dT%H:%M:%S.%
fZ')
train['title'] = title_list
train['author'] = author_list
train['domain'] = domain_list
train['tags'] = tags_list
train['length'] = train['content'].apply(len)
train['url'] = url_list

content_list, published_list, title_list, author_list, domain_list, tags_list,
url_list = extract_features(os.path.join(PATH, 'medium/test.json'))
test = pd.DataFrame()
test['content'] = content_list
test['published'] = pd.to_datetime(published_list, format='%Y-%m-%dT%H:%M:%S.%f
Z')
test['title'] = title_list
test['author'] = author_list
test['domain'] = domain_list
test['tags'] = tags_list
test['length'] = test['content'].apply(len)
test['url'] = url_list

train_target = pd.read_csv(os.path.join(PATH, 'medium/train_log1p_recommends.cs
v'), index_col='id')
y_train = train_target['log_recommends'].values

del content_list, published_list, title_list, author_list, domain_list, tags_li
st, url_list
gc.collect()
```

Out[6]: 26467

In [7]:
```
idx_split = len(train)
df_full = pd.concat([train, test])

df_full['dow'] = df_full['published'].apply(lambda x: x.dayofweek)
df_full['year'] = df_full['published'].apply(lambda x: x.year)
df_full['month'] = df_full['published'].apply(lambda x: x.month)
df_full['hour'] = df_full['published'].apply(lambda x: x.hour)
df_full['number_of_tags'] = df_full['tags'].apply(lambda x: len(x.split()))

train = df_full.iloc[:idx_split, :]
test = df_full.iloc[idx_split:, :]

train['target'] = y_train
train.sort_values(by='published', inplace=True)
train.reset_index(drop=True, inplace=True)

print('TRAIN: {}'.format(train.shape))
print('TEST: {}'.format(test.shape))
del df_full
gc.collect()
```

```
TRAIN: (62313, 14)
TEST: (34645, 13)

/home/gerardo/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:13: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy
  del sys.path[0]
/home/gerardo/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:14: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy
```

Out[7]:
```
0
```

In [8]:
```
train_df, valid_df = train[:-2000].copy(),train[-2000:].copy()
train_df.head()
```

Out[8]:

| | content | published | title | author | domain | tags | len |
|---|---|---|---|---|---|---|---|
| 0 | Susan BrattonTrusted Hot Sex Advisor To Millio... | 1970-01-01 00:00:00.001 | Saving Your Marriage By Watching Steamy Sex Ed... | Susan Bratton | medium.com | Lovemaking Sex SexPositions EarlyBird SexEdVideos | 5 |
| 1 | Ryo OoishiDec 31, 1969やってよかった中学受験明日から息子の中学受験がは... | 1970-01-01 00:00:00.001 | やってよかった中学受験 | Ryo Ooishi | medium.com | | 5 |
| 2 | なぞちゅう仮面ライダーとかスーパー戦隊を愛する30代。特撮はたしなむ程度（自称）色々なもの、... | 1970-01-18 03:21:32.400 | はてなブログに書いた今年の手帳のお話 | なぞちゅう | medium.com | 徒然日記 手帳 ブログ | 2 |
| 3 | Internet Corporation LLCDec 8, 1987SPECIAL NOT... | 1987-12-08 21:45:00.000 | Internet Corporation LLC to Acquire Early Clue... | Internet Corporation LLC | medium.com | SocialMedia EarlyClues InternetCorporationLlc | 11 |
| 4 | Mackenzie OldridgeDec 29, 2003g sowtwaretradin... | 2003-12-29 17:00:00.000 | g sowtwaretrading botMoneyMoneyMakeGetting To ... | Mackenzie Oldridge | medium.com | Finance Trading | 12 |

In [16]:
```
url2path(f'{PATH}/medium')
```

Out[16]:
```
PosixPath('/home/gerardo/fastai/fastai/../data/medium')
```

Convert your `DataFrame` in to a `DataBunch` suitable for modeling by calling `tabular_data_from_df` .

In [52]:
```
dep_var = 'target'
cat_names = ['content', 'title', 'author', 'domain', 'tags', 'url','published']
data = tabular_data_from_df(url2path(f'{PATH}/medium'), train_df, valid_df, dep
_var, tfms=[FillMissing, Categorify], cat_names=cat_names)
```

In [53]:
```
(cat_x,cont_x),y = next(iter(data.train_dl))
for o in (cat_x, cont_x, y): print(to_np(o[:15]))
```

```
[[33956 15675 17363   140 49796 38801 41361]
 [48443 27852 24466   140 38821  1194 26179]
 [54658 57644 27476   140   124 29186 41389]
 [25700 45999 13072   140 19437 41587 56699]
 [ 7819 36306  4032   140     1 13490   750]
 [58968  9225 29707   140  4186 24244 30274]
 [29396  5704 14942   140  2963 31340 53037]
 [25156 14068 12778   140 47298 44365 57594]
 [26922  2385 13606   140 10779 35466 40414]
 [19371 26644  9849   140  2872 39225 56248]
 [57532 26472 28891   140 42577 53292 36733]
 [ 7876 20957  4065   140  9576 13508  8338]
 [ 3969 29330  2051   117 30302  4309 46155]
 [45134 48427 22784   140 20690 40886 29654]
 [15447 45208  7858   140 31054 45928 10027]]
[[ 0.78925335  0.23669024  1.0345563  -0.22022635  0.9085971   1.7171918 ]
 [ 1.3236415   0.23669024 -2.0882154  -0.04523     0.9085971  -0.00542664]
 [ 1.3236415   0.23669024 -1.6197995   0.4928119   0.9085971   1.7171918 ]
 [-1.3482996   1.1729367   1.502972   -0.64300513  0.9085971  -0.2925297 ]
 [-0.8139114  -2.5720491   1.1906948   0.26505116 -2.2863035   0.8558826 ]
 [-0.8139114   0.23669024 -1.463661   -0.07531787 -0.36936316  0.5687795 ]
 [ 0.78925335  1.1729367   0.25386336  0.3600713   0.9085971  -0.57963276]
 [-0.8139114   1.1729367  -0.37069094  0.01195908  0.9085971  -0.2925297 ]
 [-1.3482996   0.23669024  0.7222791   0.09857231 -0.36936316  1.7171918 ]
 [ 0.25486508  1.1729367   0.7222791  -0.3584978   0.9085971  -0.2925297 ]
 [-0.27952313  0.23669024  0.41000193 -0.01989867  0.9085971   1.4300888 ]
 [ 0.25486508 -0.69955623  0.7222791   4.084994   -0.36936316 -0.2925297 ]
 [ 1.8580298   1.1729367   0.25386336 -0.25827864  0.9085971  -1.1538389 ]
 [ 1.3236415   0.23669024  1.3468333   0.08186913  0.9085971   0.5687795 ]
 [-0.8139114  -0.69955623  0.41000193  0.01173784 -0.36936316  0.28167644]]
[2.19722 0.69315 0.69315 3.3322  8.10198 1.38629 3.7612  3.71357 4.61512
 0.69315 4.15888 0.69315 4.89784 3.93183 5.95584]
```

Now you can create a `Learner` with `gen_tabular_dta` , and fit your model

In [55]:
```
learn = get_tabular_learner(data, layers=[200,100], emb_szs={'content': 300, 'title':300, 'author':300, 'domain':300, 'tags':300, 'url':300}, metrics=exp_rmspe)
learn.fit(1, 1e-2)
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-55-30d1cad5bed1> in <module>()
      1 learn = get_tabular_learner(data, layers=[200,100], emb_szs={'content':
300, 'title':300, 'author':300, 'domain':300, 'tags':300, 'url':300}, metrics=e
xp_rmspe)
----> 2 learn.fit(1, 1e-2)

~/fastai/fastai/basic_train.py in fit(self, epochs, lr, wd, callbacks)
    136         callbacks = [cb(self) for cb in self.callback_fns] + listify(ca
llbacks)
    137         fit(epochs, self.model, self.loss_fn, opt=self.opt, data=self.d
ata, metrics=self.metrics,
--> 138             callbacks=self.callbacks+callbacks)
    139
    140     def create_opt(self, lr:Floats, wd:Floats=0.)->None:

~/fastai/fastai/basic_train.py in fit(epochs, model, loss_fn, opt, data, callba
cks, metrics)
     89     except Exception as e:
     90         exception = e
---> 91         raise e
     92     finally: cb_handler.on_train_end(exception)
     93

~/fastai/fastai/basic_train.py in fit(epochs, model, loss_fn, opt, data, callba
cks, metrics)
     79             for xb,yb in progress_bar(data.train_dl, parent=pbar):
     80                 xb, yb = cb_handler.on_batch_begin(xb, yb)
---> 81                 loss = loss_batch(model, xb, yb, loss_fn, opt, cb_handl
er)[0]
     82                 if cb_handler.on_batch_end(loss): break
     83

~/fastai/fastai/basic_train.py in loss_batch(model, xb, yb, loss_fn, opt, cb_ha
ndler, metrics)
     21
     22     if not loss_fn: return to_detach(out), yb[0].detach()
---> 23     loss = loss_fn(out, *yb)
     24     mets = [f(out,*yb).detach().cpu() for f in metrics] if metrics is n
ot None else []
     25

~/anaconda3/lib/python3.7/site-packages/torch/nn/functional.py in cross_entropy
(input, target, weight, size_average, ignore_index, reduce, reduction)
   1644     if size_average is not None or reduce is not None:
   1645         reduction = _Reduction.legacy_get_string(size_average, reduce)
-> 1646     return nll_loss(log_softmax(input, 1), target, weight, None, ignore
_index, None, reduction)
   1647
   1648

~/anaconda3/lib/python3.7/site-packages/torch/nn/functional.py in log_softmax(i
nput, dim, _stacklevel)
   1068     if dim is None:
   1069         dim = _get_softmax_dim('log_softmax', input.dim(), _stacklevel)
-> 1070     return input.log_softmax(dim)
   1071
   1072

RuntimeError: Dimension out of range (expected to be in range of [-1, 0], but g
ot 1)
```

In [ ]:

In [56]:
```
learn
```

Out[56]:
```
Learner(data=<fastai.data.DataBunch object at 0x7f6ad2bb1400>, model=TabularMod
el(
  (embeds): ModuleList(
    (0): Embedding(60309, 300)
    (1): Embedding(59984, 300)
    (2): Embedding(30446, 300)
    (3): Embedding(219, 300)
    (4): Embedding(51584, 300)
    (5): Embedding(60302, 300)
    (6): Embedding(60223, 50)
  )
  (emb_drop): Dropout(p=0.0)
  (bn_cont): BatchNorm1d(6, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
  (layers): Sequential(
    (0): Linear(in_features=1856, out_features=200, bias=True)
    (1): ReLU(inplace)
    (2): BatchNorm1d(200, eps=1e-05, momentum=0.1, affine=True, track_running_s
tats=True)
    (3): Linear(in_features=200, out_features=100, bias=True)
    (4): ReLU(inplace)
    (5): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_s
tats=True)
    (6): Linear(in_features=100, out_features=1, bias=True)
  )
), opt_fn=functools.partial(<class 'torch.optim.adam.Adam'>, betas=(0.9, 0.99))
, loss_fn=<function cross_entropy at 0x7f6ae24c80d0>, metrics=[<function exp_rm
spe at 0x7f6adbae6840>], true_wd=True, bn_wd=True, wd=0.01, train_bn=True, path
=PosixPath('/home/gerardo/fastai/fastai/../data/medium'), model_dir='models', c
allback_fns=[<class 'fastai.basic_train.Recorder'>], callbacks=[], layer_groups
=[Sequential(
  (0): Embedding(60309, 300)
  (1): Embedding(59984, 300)
  (2): Embedding(30446, 300)
  (3): Embedding(219, 300)
  (4): Embedding(51584, 300)
  (5): Embedding(60302, 300)
  (6): Embedding(60223, 50)
  (7): Dropout(p=0.0)
  (8): BatchNorm1d(6, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
  (9): Linear(in_features=1856, out_features=200, bias=True)
  (10): ReLU(inplace)
  (11): BatchNorm1d(200, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
  (12): Linear(in_features=200, out_features=100, bias=True)
  (13): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
  (14): Linear(in_features=100, out_features=1, bias=True)
)])
```

In [ ]: