Michael Nowakowski
EECS 349: Machine Learning
Spring 2016
Professor Downey
6/8/16

ABSTRACT:

MOTIVATION:
One of the most demanding parts of filmmaking is the editing process. The editor must choose among multiple angles and line readings in order to find the one that best conveys the story while making the audience think that the events are unfolding seamlessly, that there isn't an editor at all. The motivation behind this project was to model and analyze data from a standard scene type - the two-character dialogue scene covered with traditional five-point coverage - and see if it was possible to use machine learning to generate a near-seamless edit of the scene, on par with that of the film's original editors. While this is a narrow focus relative to the entire scope of possible sequences to be editing, it is ideally one in which there is enough structure for a machine learning algorithm to begin to learn the "rules" of the art form and produce viable alternative edits.

SOLUTION:
In high-level terms, the primary focus of this project was on creating a model of scenes that offered enough information to be capable of generating new sequences of edits, based solely on the knowledge that the scene is a dialogue sequence involving two characters and that there is standard five-point coverage for the scene. The end solution developed is a list of edits for a sequence that would enable an editor to develop a sequence that closely matches the editing style of the 12 input sequences. For modeling purposes, a data set was created based on 12 scenes from well-known films (feature set included shot type, characters contained in shot, previous shot, duration of each shot in frames, timecode for start of each shot, dialogue in each snippet, and length of entire edited sequence). This model and its subsets were then used in combination with a variety of training algorithms, including Naive Bayes and MLP in Weka, the Keras library, and Andrej Karpathy's Char-RNN in order to see what could best predict and model a sequence of film edits.

TEST/TRAIN:
There were two main variations on the data set, which will be denoted as Data Set 1 and Data Set 2. The first, which was the full raw dataset, was developed by manually labeling 12 film sequences taken from *The Social Network*, *Pulp Fiction*, and *The Big Lebowski*. Each edit between clips was described as an "edit point", and consisted of the current timecode of the sequence, the duration of the clip, the clip that preceded the edit, the clip that followed the edit, the classification of each clip according to cinematic coverage conventions (i.e. closeup, medium close-up, wide, 2-shot, etc), and which characters appear in each shot. The number of edits in a scene ranged from 10 to 116, depending on length of the clip and editing style. In the end, this provided the project with roughly 500 edits categorized, and about 6,000 total values entered (future work would have to start with automated edit detection, as this is a very tedious task). An alternate version of the data set, Data Set 2, was later developed that functions much more like time-series data – each frame in the sequence is classified with a single class (a concatenation of the shot type and the character within it), and the edits are marked by changing of shots. There were tradeoffs to each approach. Each data set was separately tested using a series of techniques in search of one that might yield a good generator/predictor for future sequences based on criteria described above. For Data Set 1, test algorithms began with ZeroR to get an overview of the data set, then testing in Weka attempted to predict the Type of the clip being cut to at each edit point. The final results involved training a Char-RNN on text representations of each data set and then validating by generating brief video sequences that used the cuts specified by the algorithm to assess both how well they match standard editing techniques like shot-reverse shot (cutting back and forth between characters), consistent shot lengths ("rhythm" in editor's parlance – are the back-and-forth shots of the same length and type?) as well as a qualitative analysis of their artistic value.

RESULTS:
The Char-RNN trained on the first dataset produced very lackluster results, but the second Char-RNN, trained on a significantly simplified text version of the data set, was able to generate samples that demonstrated an adherence to the shot-reverse and rhythmic formulas of the input clips. While certainly not as capable as hoped for, the Char-RNN has learned basic editing rules and could be used as a creative tool with addition of a sizable increase in data or data aimed at replicating a more specific editing style. A video demonstrating the qualitative aspects of the edit will be available on the project website.

FULL FINAL REPORT:

Representing a Film Scene:

The most important aspect of this work was determining how to represent each clip and the sequence as a whole. A film scene, in its simplest description, is a continuous sequence of frames of video, and the frames are chosen from among several clips and laid over an audio track that is largely the same for each clip. Thus, one approach can be to assume a single audio sequence, and that video can be cut over it in any fashion desired by the editor. More on this in simplifications and assumptions. An early variation that involved classifying audio and dialogue was thrown out in favor of a simpler approach that makes many assumptions about the sequence being provided to the machine. I tried two methods to model the data, based in part on some research into representing time series data. The first approach models the events as they occur and skips all time instances where events do not occur (Data Set 1), while Data Set 2 discretizes each individual frame as a possible moment for an edit, and so encompasses the same data in a vastly different form.

Simplifications and Assumptions:

If you frequently found yourself asking "what about <insert complication here>" as you were reading the proposed solution, allow me to offer some amount of clarification here about what assumptions were made in the creation and processing of this data set. First, all the scenes used are dialogue scenes between two characters, and all were shot using traditional five(ish)-point coverage, meaning a 2-shot, two medium shots that frame characters from the waist-up, and two medium-closeups, and a few inserts that fill out the scene. All scenes are also assumed to be covered completely – e.g. every shot runs the full length of the scene and has the same exact audio track (early attempts involving use of scripts and dialogue were abandoned in favor of this significantly simpler model).

Multiple Outputs with Minimal Input

My thinking was that in order to generate new data totally independently, but of a specified length that would be usable with a scene (see “Simplifications and Assumptions” for more on how a scene is represented), I needed a method that made use of time series data and depended on past generation as input for future generation. This communication within the model keeps the machine from cutting back to the same shot or cutting to repeated angles of a single character. To do this, I felt an RNN or similar multi-output approach would be best, but I ran sample trials in Weka with basic algorithms first (see results below).

As described in the abstract, there were two variations on the data set – the more complicated Data Set 1, and the significantly simplified Data Set 2.

SAMPLE OF DATA SET 1

SCENE #	SCENE NAME	SCENE DURATION	SCENE DURATION (FRAMES)	CLIP1	CLIP 2	TIMECODE	EDIT LENGTH	CLIP2TIMECODE	CLIP2_FRAMES	CLIP2_TYPE	CLIP2_CHARACTERS
1	1-socialnetwork-bar	5:08:00	7392		1BAR	0:00:00		0	341	MED 2SHOT	1, 2
1	1-socialnetwork-bar	5:08:00	7392	1BAR	2BAR	0:14:05	0:02:05	341	53	MCU	1
1	1-socialnetwork-bar	5:08:00	7392	2BAR	3BAR	0:16:10	0:02:11	394	59	MCU	2

DATA SET 1: 12 attributes describing each “edit point” – each time the scene cut to a new shot. (~500 rows in total).

SAMPLE OF DATA SET 2:

FRAME	CLIP1_TYPE	CLIP2_TYPE
394	MCU1	MCU2
395	MCU1	MCU2
396	MCU1	MCU2
397	MCU1	MCU2
398	MCU1	MCU2
399	MCU1	MCU2
400	MCU1	MCU2
401	MCU1	MCU2
402	MCU1	MCU2

DATA SET 2: Three attributes describing each frame of each sequence separately (46,000 rows). The CLIP2_TYPE class has been updated to include characters shown, doubling the number of possible classes for CLIP2_TYPE (e.g. MCU with characters 1 and 2 becomes MCU1 and MCU2).

DATA SET 1:

Tested with 10-fold cross validation on determining CLIP2_TYPE

WEKA ALGORITHM	CORRECTLY CLASSIFIED
MLP	84.5%
Naive Bayes	79.74%

WEKA ALGORITHM	CORRECTLY CLASSIFIED
ZeroR	59.05%

DATA SET 2:

WEKA ALGORITHM	CORRECTLY CLASSIFIED
MLP	76.68%
Naive Bayes	67.40%
ZeroR	59.05%

As the results above demonstrate, Data Set 2, despite altering the representation of data drastically and increasing the number of possible CLIP2_TYPES, does not incur a significant penalty in spite of this loos. This is either due to a noisy data set or irrelevant information. The results above demonstrate that MLP can achieve nearly the same accuracy (difference of only ~7%) with this incredibly limited data set as it does with the more fully-fleshed-out Data Set 1.

RNNs and Representing Multiple Outputs:

Since future output is dependent on current output, as the data is all representing one scene-sequence, I found it useful to refer to more advanced machine learning techniques to enable generation of a multi-output sequence that received very little starting information. My first technique, which used a text representation of Data Set 1 fed into Andrej Karpathy's Char-RNN, yielded disappointing results, as the sample text size was very small. Even with parameters modified and trained on individual characters as models (a multi-hour process), the results were still very close to simply outputting the original sequence.

I found this chart to be incredibly helpful in developing an idea of representing multiple outputs and understanding RNNs.

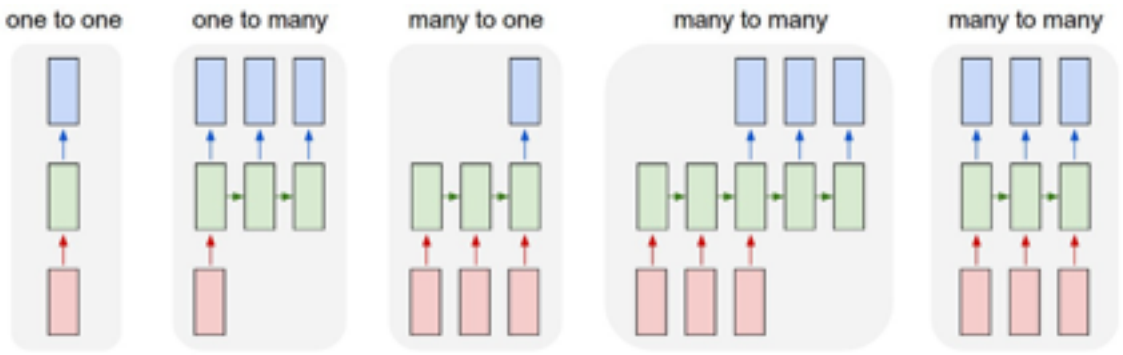

Figure 2: Karpathy's RNN Chart

The Reasonable Ineffectiveness of Andrej Karpathy's Char-RNN:

Below are results for a char-rnn trained with default settings. Regardless of temperature setting, the result is total gibberish due to the insufficient quantity of data used.

CHAR-RNN RESULTS FOR NETWORK TRAINED AS:
th train.lua -data_dir data/test -rnn_size 512 -num_layers 2

RESULTS:

```
th sample.lua cv/lm_lstm_epoch50.00_2.6790.t7 -temperature 0.3
package cunn not found!
package cutorch not found!
Falling back on CPU mode
creating an lstm...
missing seed text, using uniform probability over first character
--------------------------
MH220221                                7
                                M
                                1
                                17
                7                                               7
                                                1
                                1
1
                                23                                              1               M1
                                                                1
                                                                1

                                2
4
                                1

                2               11                                      3       7
                                                                2
1
                                2
                                                        2
                7               13                                                              1
                                                                C
                                                7
                                23      7               6                                       1
                                                                2

                                32
                1                                               11
                        7                               2                                       41
                                                2
                        3

M                               1
                                                        M
                                728
```

The Slightly Improved Efficacy of Char-RNN (Batch_Size=1)

Here are the improved results, which follow the form of the input, but never deviate from it in a dramatic fashion. In fact, most of these data points are the same as the input (again a consequence of insufficient data available).

```
(trained with batch_size=1 instead of 100)
th sample.lua cv/lm_lstm_epoch50.00_1.8799.t7 -temperature 1.0  -primetext "3    6950  MED2SHOT 3 "
package cunn not found!
```

```
package cutorch not found!
Falling back on CPU mode
creating an lstm...
seeding with 3    6950  MED2SHOT 3
--------------------------
3    6950  MED2SHOT 3
3         5647    126       MED     2
3         5647    200       MED2SHOT      3
3         5647    39        MCU     1
3         5647    54        MCU     2
3         5647    51        MCU     1
3         5647    160       MED2SHOT      3
3         5647    49        MCU     2
3         5647    128       MCU     1
3         5647    62        MED2SHOT      3
3         5647    24        MCU     1
3         5647    116       MCU     2
3         5647    200       MED2SHOT      3
3         5647    47        MCU     1
2         4770    41        CU      2
2         4770    39        CU      1
2         4770    38        MCU     2
2         4770    120       CU      1
2         4770    36        INSERT          1
2         4770    157       MED     1
2         4770    86        MED     2
2         4770    32        CU      1
2         4770    54        CU      2
2         4770    39        CU      1
2         4770    51        CU      2
2         4770    26        MCU     1
2         4770    44        CU      2
2         4770    43        CU      1
2         4770    51        CU      2
2         4770    27        MCU     1
2         4770    60        INSERT          1
```

CHAR-RNN on Data Set 2:

Here, there is a significant improvement in usable output. The char-rnn understands that frame count should increase, and it provides a good variety of shots for use in a 500 frame scene. It also has a basic understanding of shot-reverse shot – it cuts back and forth between the two characters, with a sense of editing rhythm in terms of providing relatively comparable shot lengths. One glaring failure is the duration of the MCU2 that lasts a mere 4 frames – nowhere except action movies would a 4-frame cut be acceptable to an editor.

```
th sample.lua cv/lm_lstm_epoch46.88_0.1680.t7 -temperature 1.5  -length 7000 -primetext "0,MED2
SHOT3"

Falling back on CPU mode
creating an lstm...
seeding with 0,MED2 SHOT3
--------------------------
0,MED2 SHOT3
7201,MED 2SHOT3
```

7202,MED 2SHOT3
7203,MED 2SHOT3
7204,MED 2SHOT3
7205,MED 2SHOT3
7206,MED 2SHOT3
7207,MED 2SHOT3
…
(CONDENSED REPRESENTATION BELOW)
(MED SHOT3) x 109
(MCU2) x 37
(MCU1) x 153
(MCU2) x100
(MCU1) x 16
(MCU2) x 18
(MCU1) x 34
(MCU2) x 4
(CU 2SHOT3) x 98
(MCU 2SHOT3) x 8

//For the data set below, the sequence is generated without primer text. Here again, there is an understanding of rhythm (shot and reverse shot tend to be close to the same length), and diversity of shots used is quite clear, even in this 800-frame sample.

th sample.lua cv/lm_lstm_epoch46.88_0.1680.t7 -temperature 1.0 -length 10000
Falling back on CPU mode
creating an lstm...
missing seed text, using uniform probability over first character
---
(MCU2) x 32
(MCU1) x 58
(MCU2) x 36
(MCU1) x 158
(MCU 2SHOT3) x 272
(MCU1) x 21
(MCU2) x 40
(MED1) x 64

FUTURE WORK:

Future work would involve greatly increasing the number of sequences included in the dataset, and developing the infrastructure needed to rapidly apply edit data to film sequences for qualitative evaluation.

REFERENCE:

Andrej Karpathy – “The Unreasonable Effectiveness of Recurrent Neural Networks”
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Gianluca Bontempi – “Machine Learning Strategies for Time Series Prediction”
http://www.ulb.ac.be/di/map/gbonte/ftp/time_ser.pdf