Peaches, Snakes, and Double Meanings: Applying Embeddings to Emojipastas

Aidan Casey and Simon Mason

Intro: The Problem

- Online communication often includes emojis, which are not part of most word embeddings such as word2vec
- Emojis often have double-meanings or shifting meaning
- Enter Emojipastas as seen on r/Emojipasta
    - Long form posts with a phrase-emoji pattern
    - Built mostly for humorous effect, and often includes the double-meanings emojis might contain
- The Goals:
    - Can we capture some of these aspects by training embeddings on this dataset?
    - Can this improve performance of a down stream task?

Method: Training Embeddings

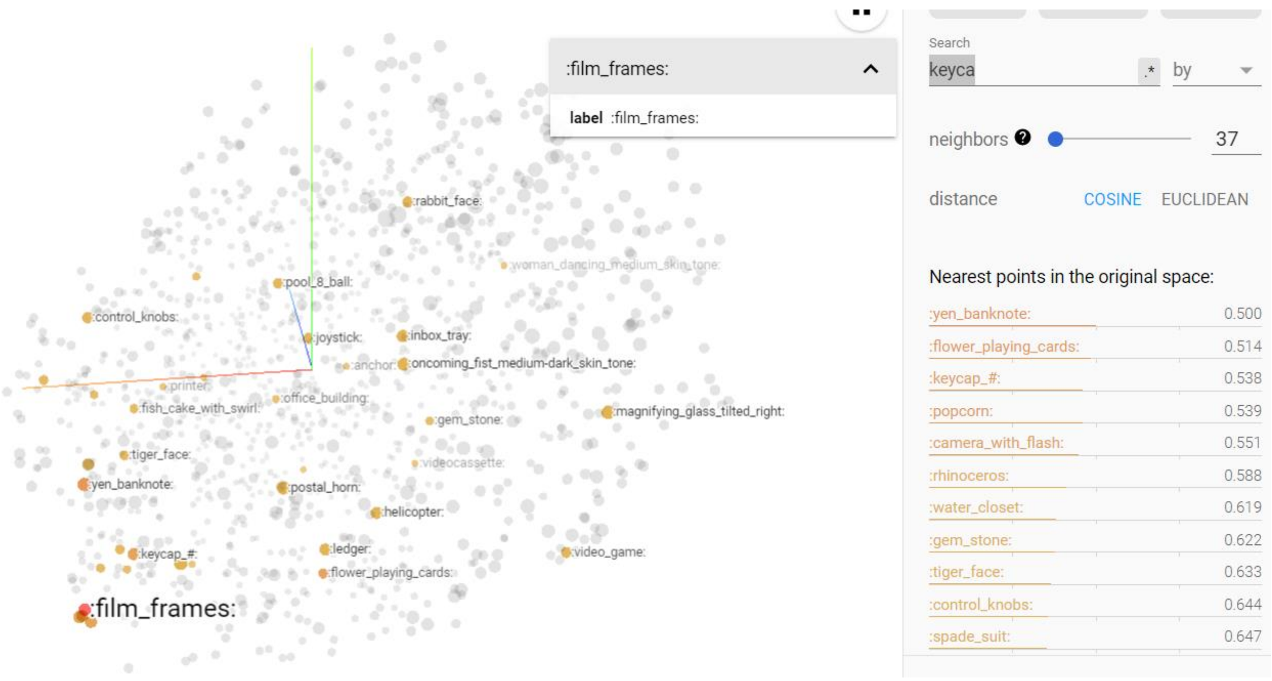
- CBOW
  - As seen in word2vec
  - Also as seen with emoji2vec, a work that created embeddings based on the official Unicode description of the emoji
  - Split data into sets of 5: target word in the middle, 4 context words on each side
  - Train a classifier based on the mean of the 4 context words
  - Use GloVe for normal English words (frozen), then start with randomized or emoji2vec embeddings for emoji and train

Method: Testing

- Visualization and Cosine Distance
  - Universal Manifold Approximation (UMAP)
- N-gram
  - Dataset seems related (phrase-emoji pattern)
  - Drawbacks: not an easy task
  - Benefits: no annotation required!

Experiments and Results: UMAP

- Cluster of Movie Related Emojis in CBOW trained emoji2vec
  - :film_frames:, :popcorn:, :camera_with_flash: are reasonably close
- Also, a keycap cluster in CBOW from randomized



:film_frames:

label :film_frames:

Search keyca .* by

neighbors 37

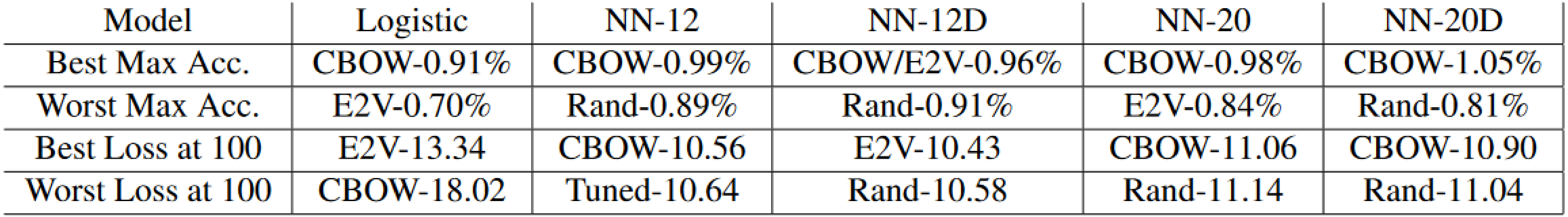
distance COSINE EUCLIDEAN

Nearest points in the original space:

:yen_banknote:	0.500
:flower_playing_cards:	0.514
:keycap_#:	0.538
:popcorn:	0.539
:camera_with_flash:	0.551
:rhinoceros:	0.588
:water_closet:	0.619
:gem_stone:	0.622
:tiger_face:	0.633
:control_knobs:	0.644
:spade_suit:	0.647

Experiments and Results: N-gram

- CBOW (from randomized) almost always did best
- Random usually did poorly
- Low accuracy and high loss, but N-gram is a very tricky task and we trained on very limited data



Model	Logistic	NN-12	NN-12D	NN-20	NN-20D
Best Max Acc.	CBOW-0.91%	CBOW-0.99%	CBOW/E2V-0.96%	CBOW-0.98%	CBOW-1.05%
Worst Max Acc.	E2V-0.70%	Rand-0.89%	Rand-0.91%	E2V-0.84%	Rand-0.81%
Best Loss at 100	E2V-13.34	CBOW-10.56	E2V-10.43	CBOW-11.06	CBOW-10.90
Worst Loss at 100	CBOW-18.02	Tuned-10.64	Rand-10.58	Rand-11.14	Rand-11.04

Discussion

- Limited data (only so much data in one subreddit)
- Emojis can have a lot of uses and most datasets are inconsistent at best (no Wikipedia filled with emoji usage)
- Emoji's can have many meanings based on context, perhaps in the future work with adapting technologies like ELMo to emojis will perform better
- Thoughts on the future:
  - New Data sources
  - New types of models for embedding and testing
  - Generative model in far future