MemeSequencer: Sparse Matching for Embedding Image Macros

Abhimanyu Dubey
Massachusetts Institute of Technology
dubeya@mit.edu

Esteban Moro
Massachusetts Institute of Technology
Universidad Carlos III de Madrid
emoro@math.uc3m.es

Manuel Cebrian
Massachusetts Institute of Technology
cebrian@mit.edu

Iyad Rahwan
Massachusetts Institute of Technology
irahwan@mit.edu

ABSTRACT

The analysis of the creation, mutation, and propagation of social media content on the Internet is an essential problem in computational social science, affecting areas ranging from marketing to political mobilization. A first step towards understanding the evolution of images online is the analysis of rapidly modifying and propagating memetic imagery or 'memes'. However, a pitfall in proceeding with such an investigation is the current incapability to produce a robust semantic space for such imagery, capable of understanding differences in Image Macros. In this study, we provide a first step in the systematic study of image evolution on the Internet, by proposing an algorithm based on sparse representations and deep learning to decouple various types of content in such images and produce a rich semantic embedding. We demonstrate the benefits of our approach on a variety of tasks pertaining to memes and Image Macros, such as image clustering, image retrieval, topic prediction and virality prediction, surpassing the existing methods on each. In addition to its utility on quantitative tasks, our method opens up the possibility of obtaining the first large-scale understanding of the evolution and propagation of memetic imagery.

KEYWORDS

image virality, image macros, feature extraction, sparse representation, embeddings, social network analysis, content understanding

ACM Reference Format:
Abhimanyu Dubey, Esteban Moro, Manuel Cebrian, and Iyad Rahwan. 2018. MemeSequencer: Sparse Matching for Embedding Image Macros. In *The 2018 Web Conference, April 23–27, 2018, Lyons, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3178876.3186021

1 INTRODUCTION

Social networks have increasingly become an integral part of modern life. Recent research in computational social science has focused on detecting the most shared content, the extent and pace of sharing of content, and the most influential content-sharing agents in a social network [6–10]. This line of inquiry, based on the study of predicting and understanding *content virality* has also risen interest in computer science [13, 52]. Content diffusion online can be understood as a product of two intertwined properties: i) the nature of the content, its evolution and mutations, and ii) the properties of the social network on which it propagates.

Diffusion of content and *cascade prediction* have received substantial attention in this domain. Several lines of recent research have focused on understanding and predicting cascades [13], the probabilities of information diffusion in cascades [47], and the recurrence of cascades [14]. These cascades are crucial in understanding the influence of the underlying social network on predicting the extent of propagation (popularity or virality) and provide strong insights into the importance of strong community structures in content propagation [54]. Extensive research has also been done in understanding the strength and extent of online community structures and their impact on information diffusion [53, 55].

With increased online big data collection and processing, research has focused on understanding content virality through the information contained in online imagery or text [5, 17, 26]. Contrary to the earlier mentioned research, this line of focus looks at the impact of content in predicting virality, independently from the network structure and its constituent effects of social reinforcement, homophily and spreading pattern. Using computer vision techniques, studies have looked at regions of images that promote content virality [20, 24].

An interesting combination of these two different lines of research is the study of evolution of information in social networks [3]. Since many memes exist in the social network that persist by mutating constantly [16, 17], understanding the mutations that are responsible for accelerating or hindering the popularity of a meme can be influential in content creation and understanding the cultural composition of online communities. An issue, however, with this line of study is the difficulty in isolating the *underlying cultural meme* from its various manifestations in online content [34, 48].

Identifying latent cultural memes from content such as tweets has been attempted first by Leskovec *et al.*[37], utilizing topic modeling, without explicitly considering mutations in content. Approaches such as n-grams and deep neural representations of text [23] have also been utilized to some success. When operating on Twitter data, hashtags provide a grounded and less noisy representation of a meme, that has been utilized in studying the propagation of associated content on social networks [42, 46]. The work of Coscia [16, 17] has studied the nature of competition between image macros online. The systematic study of mutations, however, still remains elusive under these approaches, and to the

WWW 2018, April 23–27, 2018, Lyons, France

ACM ISBN 978-1-4503-5639-8/18/04..
https://doi.org/10.1145/3178876.3186021

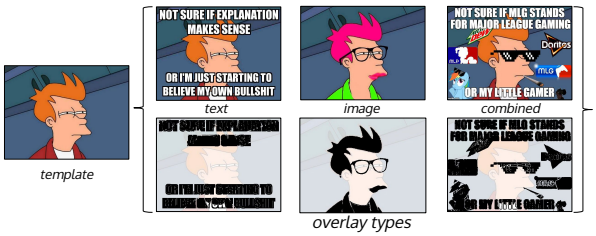

Figure 1: A sample image macro ("Futurama Fry") with its most common overlays.

best of our knowledge, there is no work on the evolution of image-based memes.

In this work, we provide a systematic framework and associated semantic feature space to study memetic imagery. Unlike the evolution of text, images mutate and evolve in a relatively controlled manner on the Internet, typical of which is the propagation of Image Macros, the most common type of online visual meme [34]. As described by Knobel and Lankshear [34], an Image Macro is the representation of an idea using an image superimposed with text optional alternative imagery. This form of representing a meme has been incredibly successful at dissemination, and is extremely popular on social networks [25].

In their most common form, Image Macros usually possess one or two lines of text flanking the template image in the center. Additionally, they may have altered imagery superimposed on the template image as well. Their etymology stems from the usage of the word "macro" in computer science, as a 'rule or a pattern that maps an input to an output' [38]. This highlights the usage of the Image Macro as a general purpose meme representation, that can be altered to fit the context specified by the overlaid text. The combination of the overarching memetic theme provided by the instantly-recognizable template image with the subtle contextual information provided by the overlaid text or imagery creates an instantly perceivable new meme that is versatile and adapted to the targeted community, justifying its prevalence in social media.

Scientific inquiry involving the propagation of these Image Macros can hence provide a stronger signal in understanding the transformation of cultural memes. The primary problem, which we focus on in this work, is the creation of a semantic representation for Image Macros that preserve the semantic information in each image macro while mapping different instances of image macros created from the same base template image close together, preserving the global context (supplied by the image), while additionally maintaining the individual context provided by each image macro. The baseline technique to solve a problem such as this would be to use deep neural network features, which have shown tremendous capabilities in encapsulating information from images. However, deep convolutional networks cannot decouple the overlaid imagery and template imagery, and process the overlay as noise, when the overlay in fact provides critical contextual information about the macro itself. This results in a loss of information from the macros, mapping most of them to the similar representation, which only amplifies as more overlays are made.

In this study, we create an algorithm that first uses the idea of sparse representation to identify template images from each Image Macro, and then using the obtained template, decouples the overlaid information from the base template. We then proceed to extract multimodal features from each image, resulting in a rich, informative and robust feature representation. Using this feature representation, we demonstrate remarkable improvements across several qualitiative and quantitative tasks involving social media imagery, demonstrating the conceptual and functional superiority of our approach from other baseline techniques. In cases where the template set is not known beforehand, we also provide an algorithm that can recover the template image from a set of sample macros based on median blending of images.

2 METHOD

Our method is based on a strong underlying assumption – memetic imagery online contains substantial amounts of Image Macros, that are constructed from a set of template images by choosing a template image and overlaying text and/or additional imagery on it. This assumption is exploited in our formulation. We begin the algorithmic description with preliminaries:

Target Set: Our target set **T** is the set of images that we wish to embed in a semantically-grounded space. In our experiments, this usually is the dataset that we conduct experiments on, scraped from websites such as Memegenerator [17] or Quickmeme [16]. We use this set of images to construct the set of template images, following Algorithm 2, that we then use for Sparse Matching and feature extraction.

Template Set: The template set **S** is the set of images with no overlays that we match each image in the target set with, to obtain the decoupled *Image Macro* representation. This template set can be supplied beforehand, but in case it is not, we construct it from the Target Set itself using an algorithm involving Sparse Matching and Median Blending.

Overlay Types: Figure 1 specifies the typical kinds of overlays on template images to produce Image Macros. The most common overlay is simple text in a white font (as shown in the section 'text overlay'). There can be modifications in the color or the addition of an image, which fall under the category of 'image overlay'. Additionally, both these things may be present together, images of which fall in the 'combined overlay' category. We take care of minute variations in color across images by contrast normalization.

The goal of our Sparse Matching algorithm is to obtain the template image any sample *Image Macro* has been constructed from. Using this template image, we can then decouple the image overlay from the Macro, and process the overlay and corresponding template separately to decouple the local context (specified by the overlay) and the global context (specified by the template).

2.1 Algorithm Overview

Our task can be summarized as learning an embedding function $f : \mathbb{R}^n \rightarrow \mathbb{R}^d$ that maps images to a low-dimensional embedding that preserves semantic content. To create an embedding, our method follows three distinct subroutines as described below:

(1) **Overlay Decoupling**: The first step of the algorithm is to identify and separate the overlaid content from the template image. To do this we employ global image contrast normalization followed

by ℓ^1-sparse reconstruction, first introduced in the seminal work on face recognition by Wright et al. [56].

(2) **Image Feature Extraction**: Image features learnt through deep convolutional neural networks (CNNs) for object classification have been shown to be tremendously powerful at capturing semantic information, and have excelled at a variety of inference tasks [21]. To capture the semantic content of the imagery, we use deep CNNs trained on image classification, and then finetuned on our target task.

(3) **Textual Feature Extraction**: To augment the information provided by the image features, we additionally extract text present in the images using optical character recognition (OCR), and augment our embedding with features extracted from this text data. To learn these text features, we use a deep recurrent neural network [39], inspired by their immense success in a variety of inference tasks in the domain of natural language processing [15, 27, 61].

After overlay decoupling and multimodal feature extraction, we concatenate the obtained text and image features to produce a powerful embedding for image retrieval and clustering of the target meme imagery, whose representational strength we verify in several experiments as described later.

2.2 Decoupling Overlays from Templates

The first subroutine of our method involves decoupling the image overlay from the template it was produced from. This can be done using a straightforward pixel-wise subtraction, however, we do not have the source template of each image *a priori*, which makes the first task the identification or matching of the correct template image from the provided test image. We begin by first normalizing the color in the image with a global pixel-wise mean normalization, to remove the slight aberrations in color across images. We then downsample each image in both the template image set $\mathbf{S}$ and target image set $\mathbf{T}$ to a fixed resolution of 48×48 pixels. Consider these downsampled sets as $\mathbf{S}_d$ (template set) and $\mathbf{T}_d$ (target set) respectively. Given these sets of normalized, downsampled images, we now describe the sparse representation algorithm.

2.2.1 Sparse Representation. Introduced by Wright et al. [56], the sparse representation algorithm provides a framework for inference under the assumption that the training samples lie on a subspace. Hence, each input test point can be written as a sparse linear combination of the training points. If we consider the training set to be the set of downsampled template meme images $\mathbf{S}_d$, and test set to be each image in the target set $\mathbf{T}_d$, we can apply the sparse representation algorithm to match each sample in $\mathbf{T}_d$ to a sample in $\mathbf{S}_d$. By this process, we can effectively recover the original template the macro was created from, and decouple the template from the overlay.

Matching: Let the total number of images present across all templates be m, and images in each class i be given by m_i. Hence, $\sum_{i=1}^{k} m_i = m$. Given a set $\mathbf{S}_{d,i}$ of m_i images, represented as a matrix $[\mathbf{s}_{1,i}, ..., \mathbf{s}_{m_i,i}] \in \mathbb{R}^{n \times m_i}$ belonging to class i, any new target sample $\mathbf{y} \in \mathbf{T}_d \subset \mathbb{R}^n$ belonging to template i will approximately lie in the linear span of the training samples of $\mathbf{S}_{d,i}$. Hence:

$$\mathbf{y} = \alpha_{1,i}\mathbf{s}_{1,i} + \alpha_{2,i}\mathbf{s}_{2,i} + \alpha_{3,i}\mathbf{s}_{3,i} + ... + \alpha_{m_i,i}\mathbf{s}_{m_i,i} \tag{1}$$

Where $\alpha_{i,j} \in \mathbb{R}$. If we consider the set of all classes $\mathbf{S}_d = \cup_i^k (\mathbf{S}_{d,i})$, we can write the matrix for all the m_i samples from each of the k classes as:

$$\mathbf{A} := [\mathbf{S}_{d,1}, \mathbf{S}_{d,2}, ..., \mathbf{S}_{d,k}] = [\mathbf{s}_{1,1}, \mathbf{s}_{2,1}, ..., \mathbf{s}_{m_k,k}] \tag{2}$$

Using this, we can write the linear representation of $\mathbf{y}$ over all training samples as:

$$\mathbf{y} = \mathbf{A}\mathbf{x}_0 \ \in \mathbb{R}^n \tag{3}$$

where, $\mathbf{x}_0 = [0, ..., 0, \alpha_{1,i}, \alpha_{2,i}, ..., \alpha_{m_i,i}, 0, ..., 0]^\top \in \mathbf{R}^m$ is a coefficient vector with zero entries except for samples belonging to class i. As proposed in [56], to obtain the sparsest solution of the above equation, we have to solve the following ℓ^0 optimization problem:

$$\hat{\mathbf{x}}_0 = \arg\min \|\mathbf{x}\|_0 \quad \text{, subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{y} \tag{4}$$

This corresponds to finding a coefficient vector $\hat{\mathbf{x}}_0$ that contains the smallest number of non-zero entries. This problem in its current form is to find the sparsest solution of an underdetermined linear system, and is NP-hard, and even difficult to approximate [4]. As proposed in [11, 22, 56], under certain assumptions of sparsity, the ℓ^0 minimization solution is equivalent to the ℓ^1 minimization solution. We therefore solve the following problem:

$$\hat{\mathbf{x}}_1 = \arg\min \|\mathbf{x}\|_1 \quad \text{, subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{y} \tag{5}$$

This corresponds to finding a coefficient vector $\hat{\mathbf{x}}_0$ that has the minimum ℓ^1-norm ($\sum_{i=1}^{m} |\mathbf{x}_i^{(i)}|$). Using standard linear programming methods, one can solve this problem in polynomial time [12]. As described in detail in [56], this algorithm recovers the correct training class even in the presence of severe occlusion (as provided by the overlaid text and/or pictures to an image), and heavy amounts of random noise. Hence, this method is ideal in our application case. Our algorithm is described in Algorithm 1. We begin by computing our downsampled template and target sets $\mathbf{S}_d$ and $\mathbf{T}_d$ respectively,

Algorithm 1: Template Matching via Sparse Representation

Input : Template Set $\mathbf{S}_d$, Target Set $\mathbf{T}_d$, threshold t_r, no. of different templates k
Output : Matched Set $\mathbf{O}_d$

Set $\mathbf{O}_d \leftarrow \emptyset$
Compute $\mathbf{A} \leftarrow [\mathbf{s}_{1,1}, \mathbf{s}_{2,1}, ..., \mathbf{s}_{m_k,k}]$
for *Image* $\mathbf{t}_i$ *in* $\mathbf{T}_d$ **do**
  $\hat{\mathbf{x}} \leftarrow \arg\min \|\mathbf{x}\|_1$, subject to $\mathbf{A}\mathbf{x} = \mathbf{t}_i$
  if $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{t}_i\|_2 \leq t_r$ **then**
    for *Class c from* 1 *to k* **do**
      Compute $z_c \leftarrow \sum_{j=1}^{m_c} \mathbb{1}\{\hat{\mathbf{x}}_{j,c} > 0\}$
    end
    Set $\mathbf{z}^{(i)} \leftarrow [z_1, z_2, z_3, ..., z_k]$
    Compute $\hat{z}^{(i)} \leftarrow \arg\max(\mathbf{z}^{(i)})$
    Set $\mathbf{O}_d \leftarrow \mathbf{O}_d \cup \{\mathbf{s}_{1,\hat{z}^{(i)}}\}$
  else
    Set $\mathbf{O}_d \leftarrow \mathbf{O}_d \cup \{\mathbf{t}_i\}$
  end
end
return $\mathbf{O}_d$

Figure 2: Sample recovery of the template images "Willy Wonka" and "Success Kid" from the target set examples. We show the obtained template at (a) 0 iterations, (b) 10 iterations, (c) 20 iterations, (d) 50 iterations and (e) 100 iterations.

and set the output set $|mathbf{O}_d$ to the empty set. For each image $\mathbf{t}_i$ in the target set, we first compute the sparse representation weights $\mathbf{x}$ as described earlier. Using the components of the sparse representation, we then proceed to evaluate the number of non-zero weights present for samples of each template class and store it in variable z_c for template class c. Owing to the sparse representation formulation, we find that only the matched template class possesses non-zero components. We then assign the template class by choosing the class with the maximum number of non-zero sparse weights, and assign the blank template image as the corrected set for the input sample $\mathbf{t}_i$. In case no matching template is found (the error in reconstruction is larger than a threshold t_r, that is $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{t}_i\|_2 > t_r$), we return the target image itself as the matched template.

Once we have the assigned template image $\mathbf{s}_{1,\hat{z}(i)}$, we can decouple the overlay from the template by computing the decoupled overlay as $\mathbf{t}_i - \mathbf{s}_{1,\hat{z}(i)}$, which can then help us do decoupled feature extraction. A caveat of this algorithm is that it requires an exhaustive template set to filter images successfully. However, in practical cases, we might not have the template set $\mathbf{S}$, in which case the next algorithm to construct the template image set $\mathbf{S}$ from the target image set $\mathbf{T}$ can be used.

Creation of Template Set: To construct a template set automatically from the target set, we describe an algorithm that utilizes the concept of median blending in image processing [59]. Median Blending is a commonly used technique in image processing to obtain a stable image from several noisy images. The central idea is to iteratively grow the template set and refine the template via successive median blending.

We begin with an empty pre-augmented template set $\mathbf{S}_{d,t}$, and iterate over the target set. For the first iteration, we simply add the input image itself to $\mathbf{S}_{d,t}$, since our template set is empty. From here on, we maintain a set of template images $\mathbf{U}_{\mathbf{t}_i}$ for every template $\mathbf{t}_i$ we identify (hence, for the first iteration, we add the first image to the set $\mathbf{U}_{\mathbf{t}_i}$). For every subsequent image, we compute the sparse coefficients $\mathbf{s}_{\hat{z}(i)}$ using $\mathbf{S}_{d,t}$, and if the input image matches with a template (even if a template is garbled, the sparse representation will ensure a sparse match, which we evaluate via the reconstruction error $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{t}_i\|_2$). If a match is found, we add the input image to the set of images corresponding to the matched template image ($\mathbf{U}_{\mathbf{s}_{\hat{z}(i)}}$). We construct the new version of the template by blending all of the images in the set of matched images corresponding to that template. This blending is done by creating a new image where each pixel is the median of the corresponding pixels in all the images

Algorithm 2: Template Set Construction

Input : Target Set $\mathbf{T}_d$, thresholds t_r, t_b
Output: Template Set $\mathbf{S}_d$

Set $\mathbf{S}_{d,t} \leftarrow \emptyset, \mathbf{S}_d \leftarrow \emptyset$
for *Image* $\mathbf{t}_i$ *in* $\mathbf{T}_d$ **do**
  Set $\mathbf{U}_{\mathbf{t}_i} \leftarrow \emptyset$
  Set $\mathbf{c}_{\mathbf{t}_i} \leftarrow 0$
  if $\mathbf{S}_{d,t} = \emptyset$ **then**
    Set $\mathbf{S}_{d,t} \leftarrow \mathbf{S}_{d,t} \cup \{\mathbf{t}_i\}$
    Set $\mathbf{U}_{\mathbf{t}_i} \leftarrow \{\mathbf{t}_i\}$
  else
    Compute $\mathbf{A}$ from $\mathbf{S}_d$ (from Algorithm 1 step 2)
    $\hat{\mathbf{x}} \leftarrow \arg\min\|\mathbf{x}\|_1$, subject to $\mathbf{Ax} = \mathbf{t}_i$
    if $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{t}_i\|_2 \leq t_r$ **then**
      Compute $\mathbf{s}_{\hat{z}(i)} \in \mathbf{S}_d$ (from Algorithm 1)
      Set $\mathbf{U}_{\mathbf{s}_{\hat{z}(i)}} \leftarrow \mathbf{U}_{\mathbf{s}_{\hat{z}(i)}} \cup \{\mathbf{t}_i\}$
      if $c_{s_{\hat{z}(i)}} = 0$ **then**
        Set $\mathbf{v} \leftarrow$ PixelWiseMedianBlending($\mathbf{U}_{\mathbf{s}_{\hat{z}(i)}}$)
        if $\|\mathbf{v} - \mathbf{s}_{\hat{z}(i)}\|_2 \leq t_b$ **then**
          Set $c_{s_{\hat{z}(i)}} = 1$
        Set $\mathbf{s}_{\hat{z}(i)} \leftarrow \mathbf{v}$
    else
      Set $\mathbf{S}_{d,t} \leftarrow \mathbf{S}_{d,t} \cup \{\mathbf{t}_i\}$
      Set $\mathbf{U}_{\mathbf{t}_i} \leftarrow \{\mathbf{t}_i\}$
    end
  end
end
for *Image* $\mathbf{s}_i$ *in* $\mathbf{S}_{d,t}$ **do**
  Set $\mathbf{S}_d \leftarrow \mathbf{S}_d \cup$ Augment($\mathbf{s}_i$)
end
return $\mathbf{S}_d$

(referred to as PixelWiseMedianBlending). For every input image, we proceed in a similar manner until the new obtained median image is within a small error of the median image in the previous iteration (we check if $\|\mathbf{v} - \mathbf{s}_{\hat{z}(i)}\|_2 \leq t_b$, if yes, we reach convergence for image $\mathbf{s}_{\hat{z}(i)}$, and set $c_{\mathbf{s}_{\hat{z}(i)}} = 1$). After convergence, we do not alter the template image. Figure 2 describes how the median image evolves with increasing iterations.

Once we have passed through all images in the target set, we augment the produced template set by random flips and crops of each template image (procedure described as 'Augment'). This is done since several Image Macros are also created from flipped and cropped versions of the template image, and this method ensures that every test image is mapped to a template correctly. This algorithm is described in Algorithm 2.

2.3 Image Feature Extraction

Once we have completed the first procedure involving decoupling the overlay, we are ready to extract features that encapsulate the semantic visual content present in the imagery. A wide variety of image features have been experimented with in computer vision literature pertaining to web and social media [20, 24, 32]. An emerging consensus in computer vision has been the extreme efficiency

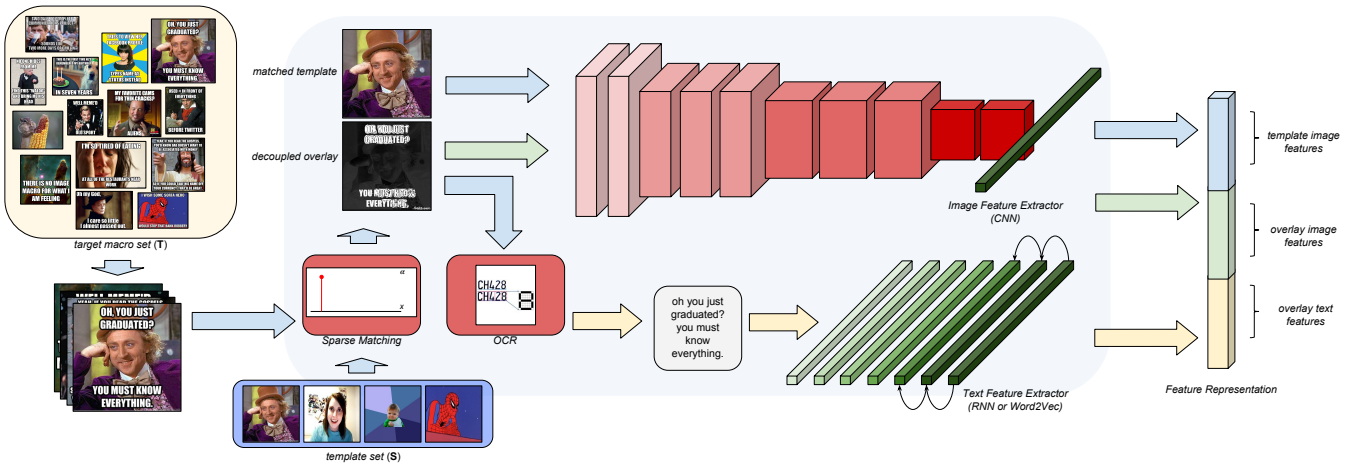

Figure 3: The central feature extraction pipeline.

of deep neural network features in capturing semantic content for classification and understanding, as described in Donahue et al. [21]. From this, we continue all image feature extraction using on deep neural network models.

2.3.1 Convolutional Neural Networks. The immense popularity of convolutional neural networks across a wide variety of computer vision tasks have made it our default choice for feature extraction. Following standard practice in computer vision [20, 24], we consider neural network models trained on the image classification dataset ImageNet [19], and additionally consider models first trained on ImageNet followed by further fine-tuning on virality prediction datasets.

2.3.2 Decoupled Feature Extraction. For each image $\mathbf{t}_i$ in the target set $\mathbf{T}$, we extract features from two separate images. We first extract features from the matched image from the template set $\mathbf{o}_i \in \mathbf{O}$, and then compute the difference image $\mathbf{d}_i = \mathbf{t}_i - \mathbf{o}_i$, and extract features from this image as well, eventually concatenating the two sets of features to form our final feature vector $\mathbf{v}_i$. Hence,

$$\mathbf{v}_i = [f(\mathbf{o}_i), f(\mathbf{t}_i - \mathbf{o}_i)] \tag{6}$$

Here, $f()$ is a function that maps an image to a multidimensional semantic feature vector, and is obtained from a CNN. Since the basic overlaying of text or additional imagery would be treated as noise by the CNN, we separate the two components and extract features separately. This ensures that images belonging to the same base template have features that are closer together in semantic space.

2.4 Textual Feature Extraction

Since most memetic imagery possess modfications in the form of overlaid text, we can exploit them to produce stronger representations by a text extraction pipeline. For this, we first run Optical Character Recognition(OCR) to obtain the text (if any) contained in the image. Following this, we extract deep neural network features based on standard practices in the field of natural language processing using deep learning [40, 58, 60].

Optical Character Recognition: We use the Tesseract [51] package for OCR, and trim excess trailing text.

2.4.1 Word2Vec Pooling. Here we extract word2vec [40] representations (following [41, 58, 60]) for each word present in the sentence, and to produce the final representation, we average over the individual word2vec representations. The word2vec implementation used is GenSim [45] with dimensionality 1000.

2.4.2 Skip-Thought Vectors. Kiros et al. [33] introduce Skip-Thought Vectors, a generic encoding scheme for sentences. The immense versatility of skip-thought vectors on a wide variety of sentence classification tasks makes them a good choice for sentence embedding creation. To extract the skip-thought (ST) features, we simply supply the extracted text to the skip-thought model, and extract the penultimate layer features.

If in the OCR phase, we do not find any text present in the image, we later replace the features with the mean text feature for that template across all images in the Target Set, in order to minimize the impact on nearest neighbor retrieval and classification. We provide ablation and comparative studies to assess the individual performance of each of the 2 abovementioned techniques in a variety of classification tasks, and find impressive results across the board, as summarized in the experiments section.

Algorithm Summary: After image correction and individual extraction of text and image features, as described in Figure 3, we obtain an informative and flexible feature representation, that preserves semantic content in the presence of overlaid text and/or imagery. We analyse the effectiveness of our obtained representation across a variety of experiments, as described in the next section.

3 EVALUATION

3.1 Experimental Setup

To showcase the effectiveness of our representation for memetic imagery, we perform a variety of qualitative and quantitative assessments. For all experiments, we use a setup involving NVIDIA TITAN

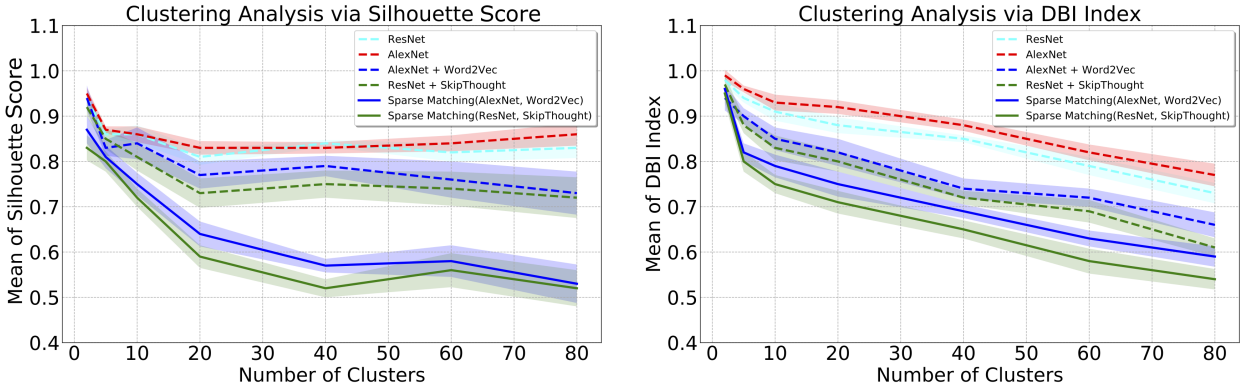

Figure 4: Evaluation of Sparse Matching on Clustering - variation of Silhouette Score (left), and Davies-Bouldin Index (right).

X GPUs, and our implementation is done using the PyTorch [43] and TensorFlow [2] frameworks. For each of the individual language and vision models, we use their publicly available weights and implementations (we use GenSim [45] for Word2Vec).

The Sparse Matching technique has two components - the image feature extractor (denoted as I), and the text feature extractor (denoted as T), giving us results of the form **Sparse Matching**(I, T) for that particular choice of feature extractors. For each experiment we conduct, we vary both the feature extractors to create different variants of the algorithm, which are all compared with (i) baseline feature extractors, (ii) previous state-of-the-art algorithms on our target tasks, (iii) each feature extractor individually and (iv) the naive combination of both feature extractors, providing an exhaustive set of comparative techniques. The CNN feature extractors we use are AlexNet [35], VGGNet-16 [49], and ResNet-18 [28]. As our results depict, the more powerful image feature extractors(ResNets and VGGNet-16) provide consistently higher performance.

The target datasets for our task are datasets pertaining to memetic imagery, typically scraped from websites such as Memegenerator, Reddit or Quickmeme. There has been interest from both the computer vision and computational social science communities in understanding such imagery, and we use the popular datasets used in these studies, with a total of 5 test splits used:
(1) **Viral Images Dataset [20]**: Introduced in [20], the Viral Images Dataset is a set of viral images filtered from the original dataset collected by Lakkaraju et al. [36] from Reddit over a period of 4 years. The dataset contains 10,078 images along with a metric for virality from 20 different image categories. This dataset contains 3 testing splits of data - (i)**Viral-Complete**(VCom), which is a collection of randomly selected pairs of images, (ii)**Viral-Pairs**(VPair), which is a collection image pairs where an image from the top 250 most-viral images is paired with an image from the top 250 least viral images, and (iii) **Viral-RandomPairs**(VRPairs), which is a set of image pairs, where one image is sampled from the top 250 most viral images, and the other is sampled at random. We utilise the predetermined training, validation and test splits.
(2) **Memegenerator Dataset [17]**: This dataset is a collection of meme images scraped from the website Memegenerator, from a period of June 27th, 2013 to July 6, 2013, accompanied with the number of upvotes for each meme as well. With a total of 326,181 images, this is the largest meme dataset, of which we use a random 70% for training, 10% images for validation and 20% for testing.
(3) **Quickmeme Dataset [16]**: Similar to the Memegenerator Dataset, this dataset is scraped from the website Quickmeme from October 2012. This dataset has a total of 178,801 images, and here as well, we use an 70-20-10 train-test-val split.
The end product of our algorithm is a versatile and efficient feature representation which can be potentially useful for several inference tasks involving online imagery, especially those that are propagated through social media, such as memes. To evalaute the efficacy of our proposed algorithm, the most natural initial experiment would be the image clustering and retrieval. A common task online would be phrased as "given a certain test image $\mathbf{I}$ and image set $\mathcal{S}_\mathbf{I}$, what are the images in $\mathcal{S}_\mathbf{I}$ that are most similar (in semantic content) to the test image $\mathbf{I}$?", and indeed, this is the first target task we design our algorithm for, and to evaluate our algorithm on this task, we look at three separate qualitative and quantitative experiments, as summarized below:

3.2 Image Clustering and Retrieval

Clustering Analysis: A summarizable version of the retrieval task is the task of image clustering - understanding how our feature representation embeds the space of images. Considering that these images form smaller subsets in semantic space, we would expect distinct clusters being formed in the embedding space. Since we do not necessarily have labels for classes in most of our datasets, to assess the quality of clustering we first compare the *internal clustering metrics* for our representation with the existing state-of-the-art and baseline methods. The Silhouette Score [44](SS) and the Davies-Bouldin Index [18](DBI) are two popular internal metrics used to quantifiably determine the quality of clustering of a representation. We proceed by clustering points from the Memegenerator Dataset [16] and vary the number of desired clusters, to obtain the clustering indices for each feature extraction algorithm as functions of the number of clusters required. Here, we use the clustering algorithm K-means, with an off-the-shelf implementation in Scipy [31].

The desiderata for efficient clustering are separable clusters, with no significant overlap, and high similarity within clusters. Or, put succintly, high intra-cluster similarity, and low inter-cluster similarity. If we consider the Silhouette Score (SS), for clusters that

are balanced in size and fairly separable, we desire a low mean value of SS across clusters, and a low deviation of SS across clusters as well - representing the average deviation of cluster items from the cluster mean. Figure 4a summarizes the results for SS analysis, which demonstrates the distinctive clustering our algorithm provides. Similarly, for the metric DBI (Figure 4b), we consider the Euclidean distance as the distance metric, and observe consistently lower values, consolidating the efficiency of our algorithm in creating rich feature representations.

Image Retrieval and Visualization: The previous set of results provide a quantiative summary of the improved clustering provided by our algorithm, but do not make precise an intuition to validate our representation. To strengthen the qualitative intuition behind the performance of our algorithm, we visualize using a nearest neighbor retrieval.

We provide a set of example image retrieval queries, and compare the retrieved images for the nearest neighbours of the query in feature space. As summarized in Figure 5, the sparse matching algorithm is robust to changes in text information, and by decoupling the text, base template and overlaid imagery, we see a richer embedding that preserves these changes in semantic space, whereas other algorithms simply treat an aberration or overlaid text as noise, providing poorer retrieval results. Just simply using image features retrieves visually similar images, yet, their text content is dissimilar. Using simply text features proceeds with the reverse effect, with various imagery being returned. Our method, however, returns a semantically aligned set of images.

Observing the robustness of our representation and the qualitiative improvements in image retrieval, a natural extension would be understanding the performance of the feature representation on more complex tasks involving online imagery, such as inferring the popularity, content and timeline of an image. These quantitative evaluation tasks are motivated by several observations on online imagery and online content itself. It has long been established in social science literature that memetic content on the internet is ephemeral and rapidly evolving [20, 24, 36], which makes the challenge of predicting the nature of content and the timeline of its existence an interesting problem. Building on this line of thought, we now describe the quantitative experiments we perform to assess our algorithm.

3.3 Quantitative Evaluation

We perform a series of varied inference tasks to assess the quality of representations generated by our algorithm. The general framework for evaluation in these experiments is as follows - (i) we first select a CNN-based image feature extractor from the set of CNNs described earlier, and select one of two text feature extractors, (ii) the CNN feature extractor is fine-tuned naively on the target task using the assigned training set, and (iii) we extract both image features and text features from the designated neural networks, following the Sparse Matching algorithm. Finally, once the features are obtained, we train a multiclass SVM classifier [29] for each of the following inference tasks. If the task is a ranking task (as Virality Prediction), we train a RankSVM [30] instead.

Topic Prediction: The Viral Images Dataset [20] is constructed of images scraped from Reddit [1], along with the subreddits these images were posted to. The subreddits are curated lists belonging to a particular genre or topic of content, and example subreddits are `/r/funny`, `/r/aww`, `/r/wtf`, `/r/gaming` and `/r/atheism`, and the resulting inference task is that of predicting the correct topic or subreddit the image was posted to.

Algorithm	Accuracy(%)	
	Topic	Timeline
SVM + Image Features [20]	41.09	24.14
SVM + 2x2 Dense HOG Features	43.08	25.18
Finetuned AlexNet [35]	53.89	32.15
Finetuned VGGNet16 [49]	57.21	33.02
Finetuned ResNet18[49]	58.17	35.16
SVM + Word2VecPooling [40]	55.19	18.07
SVM + SkipThought Features [33]	58.81	19.15
SVM + ResNet18 + Word2VecPooling [40]	65.35	32.15
SVM + ResNet18 + SkipThought Features [33]	69.18	34.06
Xiao and Lee [57]	70.57	38.18
Singh and Lee [50]	71.85	37.09
Dubey and Agarwal [24]	75.57	40.17
Sparse Matching (AlexNet, Word2VecPooling)	72.18	37.15
Sparse Matching (VGGNet16, Word2VecPooling)	76.08	39.57
Sparse Matching (ResNet18, Word2VecPooling)	77.85	40.88
Sparse Matching (AlexNet, SkipThought)	74.23	43.44
Sparse Matching (VGGNet16, SkipThought)	78.98	45.78
Sparse Matching (ResNet18, SkipThought)	**80.69**	**46.91**

Table 1: On both tasks of topic and timeline prediction, we observe that Sparse Matching provides substantial improvements in performance.

The task is challenging primarily owing to the overlap in imagery across categories, and the overlap in context as well. For example, several categories may contain images of different animals or people, as well as a significant overlap in the template images that might be used. We compare our performance on this task with several benchmark and preivous state-of-the-art methods on this task, summarized in Table 1. Since there are 20 different categories, random chance performs at 5%. Due to the immense overlap in content, naive algorithms perform poorly, with less than 50% prediction accuracy.

By just considering the text content itself, we see a slight increase in performance, which increases further still when both feature modalities are combined (image CNN is fine-tuned). However, as hypothesized, these features are corrupted by the coupling of text and image content, and hence perform averagely. Approaches that take context into account, such as the work of Singh and Lee [50], and Dubey and Agarwal [24] perform much better, however, they face challenges in decoupling text information. Finally, we see that Sparse Matching performs substantially better, with a maximum gain (with ResNet18 visual features and Skip Thought language features), of 5.12% in performance.

Timeline Prediction: The results of the preivous experiment provide us with evidence regarding the representational power of our method. An extension of the first experiment would be to predict, along with the topic of the content, the time period it was posted in. Applications of such a task are present in content creation, and prediction of popularity trends in social media. Since

Algorithm	Percentage Accuracy on Dataset				
	VCom[20]	VPairs[20]	VRPairs[20]	MemeGenerator[16]	QuickMeme[17]
RankSVM + Image Features [20]	53.40	61.60	58.49	59.12	57.05
RankSVM + 2x2 Dense HOG Features	52.75	58.81	56.92	58.87	55.56
RankSVM + AlexNet fc7 Features [35]	54.41	61.93	58.58	59.91	58.03
RankSVM + VGGNet-16 fc7 Features [49]	55.18	63.01	59.15	60.12	61.12
RankSVM + Word2VecPooling [40]	60.11	61.78	59.94	61.49	62.02
RankSVM + SkipThought [33]	63.06	64.12	60.23	65.57	64.28
RankSVM + ResNet18 + Word2VecPooling [40]	66.05	70.98	70.33	71.06	69.45
RankSVM + ResNet18 + SkipThought [33]	69.36	74.51	72.09	75.53	73.81
Xiao and Lee [57]	63.25	75.23	73.22	70.11	71.21
Singh and Lee [50]	65.87	76.20	74.38	72.25	70.08
Dubey and Agarwal [24]	68.09	78.38	76.95	74.43	74.51
Sparse Matching (AlexNet, Word2VecPooling)	70.02	82.21	80.04	79.53	79.95
Sparse Matching (VGGNet16, Word2VecPooling)	70.93	83.03	81.15	79.94	80.22
Sparse Matching (ResNet18, Word2VecPooling)	71.87	84.19	81.96	80.01	80.27
Sparse Matching (AlexNet, SkipThought)	70.86	83.05	81.29	80.25	80.16
Sparse Matching (VGGNet, SkipThought)	71.54	83.98	82.34	81.16	80.87
Sparse Matching (ResNet18, SkipThought)	**73.03**	**85.15**	**82.62**	**81.80**	**80.9**1

Table 2: Comparison of performance on the task of virality prediction. We observe that with our feature representation performance is unmatched compared to the existing state-of-the-art, by a significant margin.

content on viral image websites and memes are continually evolving, we see that the content alone can be a strong indicator of the time period it represents, as displayed by our prediction results in Column 2 of Table 1.

We observe that naive baselines for both basic image features and text features fail miserably at this task, capturing very little information about the context of an image, crucial for a task like timeline prediction. Deep image features perform slightly better, and a combination of deep image features and text features provides good performance, which is ousted by previous virality prediction techniques, and Sparse Matching. The decoupled context provides a signficant boost over the previous state-of-the-art, with an improvement of **6.74%**.

Virality Prediction: With the impressive increase in performance on timeline prediction, we have evidence supporting the improved ability of our method to identify temporal distinctions in memetic imagery. This demonstration leads us to our final quantitative experiment, virality prediction. As summarized by recent work on detection of viral content and cascades in social networks [36], capturing virality is a difficult problem, with challenges arising from the presence short-lived content, rapidly evolving diverse imagery. With our model, however, we hypothesize that predicting virality would be possible with greater efficiency owing to the stronger, robust representation.

Hence, we evaluate the prediction performance on the primary inference task of virality prediction, following the methodology introduced in [24]. Here, the prediction of virality is taken to be a pairwise classification task, where given an input pair of images, the algorithm needs to predict the more viral image of the two. This evaluation scheme has been shown to be robust to network effects [24], and is representative of evaluation of the image virality based on the content alone. We follow the same procedure, replacing our SVM classifier with a Ranking SVM [30] classifier to learn a function that provides pairwise predictions. We evaluate our algorithm on 5 different datasets - the 3 test splits of the Viral Images Dataset [20] - (i) Viral-Complete, (ii) Viral Pairs, and (iii) Viral-RandomPairs, and the Memegenerator [16] and Quickmeme datasets [17]. For all experiments where a train-test split is not predefined, we use a 70-20-10 (train-test-val) split, following standard protocol.

Our experiment is summarized in Table 2. Note here that since this is a pairwise prediction task, random chance performs at 50%. Again, as seen in the previous experiments, naive image-based algorithms perform only slightly better than random chance, with accuracies around 50-60%. Operating only on extracted text data gives us slightly better performance, but it fails to generalize since a lot of imagery do not possess any relevant text at all, which provides the algorithms with only noise. Sparse Matching provides large improvements, as high as **7.37%** in some cases. Even the weakest versions of Sparse Matching perform **3.8%** better than the previous state-of-the-art.

Finally, we attempt to analyse the durability of our method in virality prediction. Since the nature of viral content changes rapidly over time, we would want to learn a classifier that can perform well for longer periods of time, given an input training set taken at any prior time. We would not want to keep re-training our algorithm often as new data keeps coming in, and reduce the time spent updating our model. To ascertain the invariance of methods in such a manner, we devise a final quantitative experiment.

Temporal Robustness Analysis: To examine the resilience of our algorithm to evolving data, we devise the following experiment: The Memegenerator dataset [16] contains images from 13 different time periods, as described earlier. We select data from the i^{th} time period as our training data, and test the trained algorithm on the $k-i$ sets of test images from subsequent time periods. We perform $\sum_{i=1}^{k-1}(k-i) = \frac{k(k-1)}{2}$ total experiments, which in our case is 78. We report the average obtained performance for each algorithm, summarized in Table 3.

Figure 5: 4-Nearest neighbour image retrieval results for two sample queries.

Algorithm	Accuracy(%)
SVM + Image Features [20]	51.02
SVM + 2x2 Dense HOG Features	51.18
Finetuned AlexNet [35]	51.75
Finetuned VGGNet16 [49]	51.56
Finetuned ResNet18 [49]	51.97
SVM + Word2VecPooling [40]	52.02
SVM + SkipThought Features [33]	52.95
SVM + Finetuned ResNet18 + Word2VecPooling [40]	55.68
SVM + Finetuned ResNet18 + SkipThought Features [33]	56.35
Xiao and Lee [57]	62.67
Singh and Lee [50]	61.98
Dubey and Agarwal [24]	62.83
Sparse Matching (AlexNet, Word2VecPooling)	61.78
Sparse Matching (VGGNet16, Word2VecPooling)	61.97
Sparse Matching (ResNet18, Word2VecPooling)	63.49
Sparse Matching (AlexNet, SkipThought)	64.07
Sparse Matching (VGGNet16, SkipThought)	64.18
Sparse Matching (ResNet18, SkipThought)	**65.12**

Table 3: Evaluation of Sparse Matching on the Temporal Robustness task.

It is vital to understand the difficulty of this task compared to the previous virality prediction experiment. For each time period, we have (on average) only 7% of the previously available training data, which provides a very sparse signal. This is consistent with the observation that most methods (including naive deep learning algorithms) perform very close to random chance, given the lower correlation between the observed training data and the test data. We still observe, however, that specialized methods perform much better than random chance, and Sparse Matching provides the best performance, showcasing again the richness and versatility of the method in capturing context.

4 DISCUSSION AND FUTURE WORK

In this work, we introduced the task of understanding the creation, mutation and propagation of image macros online as a representation for the growth and evolution of a meme. We introduced a method that succesfully maps any arbitrary image macro to the template it was created from, and then embeds it into a powerful semantic space. We demonstrated the robustness and richness of the semantic embedding space via a variety of relevant experiments on online imagery - image retrieval, clustering, content analysis, popularity analysis and temporal evolution, all with remarkable performance.

Our work is the first of its kind in the domain of web content analysis that looks at the virality prediction problem through the lens of the Image Macro structure, and successfully exploits it. While prior work in the domain of virality prediction from content has focused on the originality of memes [17], popularity-promoting regions within meme images [24], relative attributes and category analysis [20], our approach provides a technique to systematically study the evolution of ideas and memes online.

As a demonstration of this technique, we provide a sample phylogenetic tree constructed from the pairwise distances between different image macros, obtained using our algorithm (see Figure 6). This tree displays the mutation of memes from a popular template, created using MATLAB's `seqlinkage` command (the figure displays selected nodes). We see that our embedding is powerful enough to capture subtle evolutionary patterns in the mutation of the meme, with children being semantically linked either through imagery (note the subtree with the hat overlay), or through text (subtree with pokemon text). With large amounts of time-series meme imagery, our embedding technique paves way for the study of systematic evolution of ideas on the Internet.

All previous efforts in the study of image virality have been futile at extracting these subtle variations in memes, and creating evolutionary trees such as the one displayed, despite having success at the task of predicting virality [20, 24]. Our contribution through this work enables the large-scale study and analysis of such evolutionary trends online, with numerous applications in combining the effects of content and network structure in understanding information diffusion and evolution. Such a study can be influential in creating methods for identifying original content and their sources, and creating a robust science for understanding how multimedia propagates on the Internet.

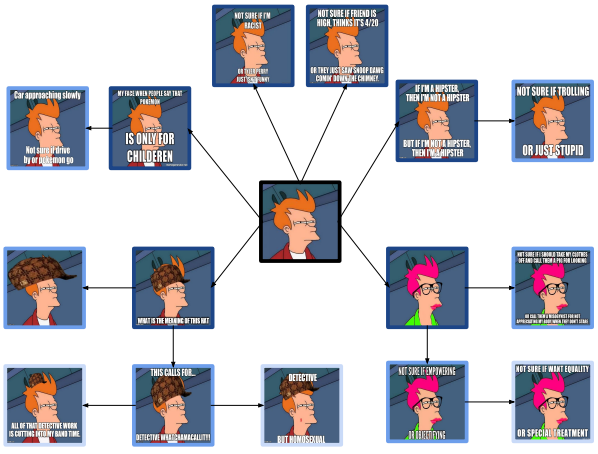

Figure 6: Pruned phylogenetic tree constructed from 1 meme template and its neighbours using distances generated by Sparse Matching.

REFERENCES

[1] [n. d.]. reddit: the front page of the internet. http://www.reddit.com/. ([n. d.]). Accessed: 2017-09-30.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. [n. d.]. TensorFlow: A System for Large-Scale Machine Learning.

[3] Lada A Adamic, Thomas M Lento, Eytan Adar, and Pauline C Ng. 2016. Information evolution in social networks. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 473–482.

[4] Edoardo Amaldi and Viggo Kann. 1998. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science* 209, 1-2 (1998), 237–260.

[5] Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. 2012. The role of social networks in information diffusion. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 519–528.

[6] Albert-László Barabási. 2016. *Network science*. Cambridge university press.

[7] Jonah Berger and Katherine L Milkman. 2012. What makes online content viral? *Journal of marketing research* 49, 2 (2012), 192–205.

[8] Jonah Berger and Eric M Schwartz. 2011. What drives immediate and ongoing word of mouth? *Journal of Marketing Research* 48, 5 (2011), 869–880.

[9] Vincent Buskens and Jeroen Weesie. 2000. Cooperation via social networks. *Analyse & Kritik* 22, 1 (2000), 44–74.

[10] Vincent Buskens and Kazuo Yamaguchi. 1999. A new model for information diffusion in heterogeneous social networks. *Sociological methodology* 29, 1 (1999), 281–325.

[11] Emmanuel J Candes, Justin K Romberg, and Terence Tao. 2006. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics* 59, 8 (2006), 1207–1223.

[12] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. 2001. Atomic decomposition by basis pursuit. *SIAM review* 43, 1 (2001), 129–159.

[13] Justin Cheng, Lada Adamic, P Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. 2014. Can cascades be predicted?. In *Proceedings of the 23rd international conference on World wide web*. ACM, 925–936.

[14] Justin Cheng, Lada A Adamic, Jon M Kleinberg, and Jure Leskovec. 2016. Do Cascades Recur?. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 671–681.

[15] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, 160–167.

[16] Michele Coscia. 2013. Competition and Success in the Meme Pool: A Case Study on Quickmeme. com.

[17] Michele Coscia. 2014. Average is boring: How similarity kills a meme's success. *Scientific reports* 4 (2014), 6477.

[18] David L Davies and Donald W Bouldin. 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence* 2 (1979), 224–227.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 248–255.

[20] Arturo Deza and Devi Parikh. 2015. Understanding image virality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1818–1826.

[21] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*. 647–655.

[22] David L Donoho. 2006. For most large underdetermined systems of linear equations the minimal ÃřÂİÂŞÂĄ1-norm solution is also the sparsest solution. *Communications on pure and applied mathematics* 59, 6 (2006), 797–829.

[23] Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts.. In *COLING*. 69–78.

[24] Abhimanyu Dubey and Sumeet Agarwal. 2017. Modeling Image Virality with Pairwise Spatial Transformer Networks. In *Proceedings of the 2017 ACM on Multimedia Conference, MM 2017, Mountain View, CA, USA, October 23-27, 2017*. 663–671. https://doi.org/10.1145/3123266.3123333

[25] Phil Edwards. [n. d.]. The reason every meme uses that one font. https://www.vox.com/2015/7/26/9036993/meme-font-impact/. ([n. d.]). Accessed: 2017-09-30.

[26] James P Gleeson, Kevin P OâĂŹSullivan, Raquel A Baños, and Yamir Moreno. 2016. Effects of network structure, competition and memory time on social spreading phenomena. *Physical Review X* 6, 2 (2016), 021019.

[27] Alex Graves and Jurgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18, 5 (2005), 602–610.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[29] Chih-Wei Hsu and Chih-Jen Lin. 2002. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks* 13, 2 (2002), 415–425.

[30] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 133–142.

[31] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001–. SciPy: Open source scientific tools for Python. (2001–). http://www.scipy.org/ [Online; accessed 9/30/2017].

[32] Aditya Khosla, Atish Das Sarma, and Raffay Hamid. 2014. What makes an image popular?. In *Proceedings of the 23rd international conference on World wide web*. ACM, 867–876.

[33] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. 3294–3302.

[34] Michele Knobel and Colin Lankshear. [n. d.]. Online memes, affinities, and cultural production. ([n. d.]).

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[36] Himabindu Lakkaraju, Julian J McAuley, and Jure Leskovec. 2013. What's in a Name? Understanding the Interplay between Titles, Content, and Communities in Social Media. (2013).

[37] Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 497–506.

[38] M Douglas McIlroy. 1960. Macro instruction extensions of compiler languages. *Commun. ACM* 3, 4 (1960), 214–220.

[39] LR Medsker and LC Jain. 2001. Recurrent neural networks. *Design and Applications* 5 (2001).

[40] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[41] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696* (2016).

[42] Ruth Page. 2012. The linguistics of self-branding and micro-celebrity in Twitter: The role of hashtags. *Discourse & communication* 6, 2 (2012), 181–201.

[43] Adam Paszke, Sam Gross, and Soumith Chintala. 2017. PyTorch. (2017).

[44] Slobodan Petrovic. 2006. A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters. In *Proceedings of the 11th Nordic Workshop of Secure IT Systems*. 53–64.

[45] R Rehurek and P Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3, 2 (2011).

[46] Daniel M Romero, Brendan Meeder, and Jon Kleinberg. 2011. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In *Proceedings of the 20th international conference on World wide web*. ACM, 695–704.

[47] Kazumi Saito, Ryohei Nakano, and Masahiro Kimura. 2008. Prediction of information diffusion probabilities for independent cascade model. In *Knowledge-based intelligent information and engineering systems*. Springer, 67–75.

[48] Limor Shifman. 2013. Memes in a digital world: Reconciling with a conceptual troublemaker. *Journal of Computer-Mediated Communication* 18, 3 (2013), 362–377.

[49] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[50] Krishna Kumar Singh and Yong Jae Lee. 2016. End-to-end localization and ranking for relative attributes. In *European Conference on Computer Vision*. Springer, 753–769.

[51] Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, Vol. 2. IEEE, 629–633.

[52] Thomas W Valente. 1995. Network models of the diffusion of innovations. (1995).

[53] Lilian Weng and Filippo Menczer. 2015. Topicality and impact in social media: diverse messages, focused messengers. *PloS one* 10, 2 (2015), e0118410.

[54] Lilian Weng, Filipspo Menczer, and Yong-Yeol Ahn. 2013. Virality prediction and community structure in social networks. *Scientific reports* 3 (2013), 2522.

[55] Lilian Weng, Filippo Menczer, and Yong-Yeol Ahn. 2014. Predicting Successful Memes Using Network and Community Structure.. In *ICWSM*.

[56] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. 2009. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence* 31, 2 (2009), 210–227.

[57] Fanyi Xiao and Yong Jae Lee. 2015. Discovering the spatial extent of relative attributes. In *Proceedings of the IEEE International Conference on Computer Vision*. 1458–1466.

[58] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical Attention Networks for Document Classification.. In *HLT-NAACL*. 1480–1489.

[59] Chenxi Zhang, Jizhou Gao, Oliver Wang, Pierre Georgel, Ruigang Yang, James Davis, Jan-Michael Frahm, and Marc Pollefeys. 2014. Personal photograph enhancement using internet photo collections. *IEEE transactions on visualization and computer graphics* 20, 2 (2014), 262–275.

[60] Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820* (2015).

[61] Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep Learning for Chinese Word Segmentation and POS Tagging.. In *EMNLP*. 647–657.