Stonkinator: An Automatic Generator of Memetic Images

José P. Lopes, João M. Cunha and Pedro Martins
University of Coimbra,
Centre for Informatics and Systems of the University of Coimbra, Department of Informatics Engineering
{joselopes,jmacunha,pjmm}@dei.uc.pt

Abstract

In an increasingly active and global society, memes came to be seen as a means of communication and entertainment in digital culture. Throughout the years, many image-based memes have been produced to be used in computer-mediated communication. In these images, semiotics play an important role, often involving processes of citation, parody, remix, among others. In this paper, we present a system that generates *memes* in the *Stonks* format, using a sentence introduced by the user as input. With this system, which we called *Stonkinator*, we aim to help users create their own memes. To achieve this, the system generates memes with a single input, using techniques of image analysis and blending. The *Stonkinator* system was tested by 23 participants and the results show that the system is able to produce a wide variety of memes, which, according to participants, can be used on social networks and in informal text conversations.

Introduction

Dawkins (1976) coined the term *meme* (from the Greek word *mimema*) to describe a transmission unit that holds an idea or behaviour between human beings, crossing generational and cultural barriers. Nowadays, the term is often associated with images and videos from the internet, usually containing humorous content, whose objective is to transmit an idea or message with the ability to cross language and cultural barriers (Shifman 2013). These image-based memes are greatly used in computer-mediated communication, being rapidly created and spread, and ultimately playing an important role in digital culture (Wiggins 2019).

The growing use of memes (Kostadinovska-Stojchevska and Shalevska 2018) has led to the development of multiple tools that help to create memes, e.g. Imgflip's MemeGenerator. Existing meme-generation systems usually allow the user to produce multiple types of output, making it difficult to determine the *genre* to which the meme belongs and understand how it should be used. In addition, different meme templates exist and not all have the same degree of complexity: some are based on a structure that has always the same background image and only the text is changed; others have a structure in which both the imagery and text are changed. In the field of Computational Creativity, there are already systems that explore meme generation related to the former kind, e.g. Oliveira, Costa, and Pinto (2016) generate memes by finding the most suitable meme template for a given news headline. The latter kind involves multimodal exploration, requiring both text and image to be changed. We consider that computational creativity techniques can be especially useful for this kind of meme generation, for example by using *conceptual extension* and *visual blending* (Cunha 2022; Cunha, Martins, and Machado 2020).

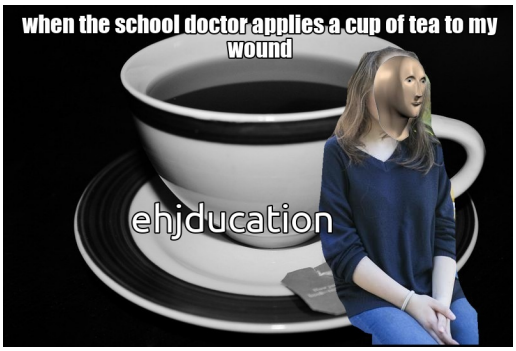


Figure 1: Meme generated by *Stonkinator* with the input "When the school doctor applies a cup of tea to my wound".

In this paper, we focus on *Stonks memes*, which are often used as reaction images, meant to portray a specific emotion in response to something that has been said, and represent an excessive proud feeling over the completion of a simple task ("Adam" and "Don" 2020). Our goal is to facilitate the creation of *Stonks memes* (Fig. 1) to be used in social media and instant message conversations. We present *Stonkinator*, a tool that takes a sentence or expression as input and uses it to generate memes in the Stonks format (Fig. 2). The *Stonkinator* system is divided into four modules: (i) the text handler; (ii) the image obtainer; (iii) the image analyser; and (iv) the image blender. The first module analyses the input sentence given by the user and extracts a theme or subject, which is sent to the second module. The second module obtains a set of images related to the theme identified by the first module. The third module analyses the images and selects one containing a person and another one to be used as the background. It also resizes the images and creates a mask image to be used in the blending process. Lastly, the

fourth module receives the image containing the person, the background and the mask and performs the blending. It also writes the sentence used as input, the retrieved word related to the action, and places the Stonks character's head in the place of the person's head.

For the implementation of *Stonkinator*, we used different types of blending. In order to assess the preferred type of blending and if the users would have an interest in sharing the output obtained, we conducted a user study with 23 participants. The results show a preference for images created using a simple pasting method and positive feedback in terms of usage in private conversations and sharing on social networks. For testing purposes, *Stonkinator* was adapted to be used as a website, which we plan to make available in the future.

Overall, we consider that the presented project has the following positive aspects:

- efficiency: there's no need to manually analyse, select and segment image elements.
- diversity: by scraping multiple images related to the input, the system is able to return different results each time it runs, whether it's a different retrieved word, background or person in the output;

The remainder of this paper is organised as follows: the second section presents related work to memes and visual blending, the third section describes the *Stonkinator* framework, the fourth section describes an experiment and analyses the results obtained and the fifth section shows the improved work done to the tested version. Lastly, the sixth section presents the final conclusions and future directions.

Related Work

In this section, we introduce projects that have been a source of inspiration for the development of our work and explain how they relate to the *Stonkinator* system. We first describe systems related to meme generation then existing work on visual blending.

Memes

Due to the growing use of memes (Kostadinovska-Stojchevska and Shalevska 2018), multiple tools have been developed to help to create these images. Imgflip's MemeGenerator[1] provides a wide array of commonly used images to choose from and lets the user write their own captions with different settings, such as font family, font size, position, and more.

Another example is Imgflip's AI Memes,[2] which lets the user choose an image and generates a caption using a deep artificial neural network. A different approach is used by AI-Memes,[3] in which the user inputs a search query and is presented with 10 meme image choices that match the search criteria, from which the user selects one. Then, the AI system generates another 10 potential captions for the selected image, and the user chooses their preferred caption for the meme.

Figure 2: Stonks meme

Different *meme templates* exist (Nissenbaum and Shifman 2018), e.g. the *Crying Peter Parker*, and these can also be divided into different *meme genres*, which are used in different situations (Shifman 2013). For example, one of the meme genres used in online forums and chats is called *reaction shots*, which are used to transmit an emotion or reaction to the addressee of the meme (Milner 2012). Existing meme-generation systems usually allow the user to produce multiple types of output, making it difficult to determine the *genre* to which the meme belongs and understand how it should be used.

Moreover, not all meme templates have the same degree of complexity: some always use the same background image and only the text is changed; others have a structure in which both the imagery and text are changed. In the field of Computational Creativity, there are already systems that explore meme generation in both formats. Oliveira, Costa, and Pinto (2016) focus on the automatic generation of internet memes, based on macro-images, that is, potential combinations between an image and a text, in order to be spread on social networks. Memes are produced from news headlines, to which, according to linguistic characteristics, certain macro-images are associated and the text is adapted according to the meme template. Another example is the work by Sadasivam et al. (2020), which also takes advantage of these templates. The authors present an automatic meme generator that creates memes based on a textual input and the system combines macro-images with text caption, using an encoder-decoder model to produce the final image. Other meme generators include the work by Lin et al. (2021), which addresses the problem of meme generation as an image captioning task by using an encoder–decoder architecture to generate Chinese meme texts that match image content. Miliani et al. (2020) propose a shared task for automatic classification of internet memes which includes meme detection, hate speech identification and event clustering. Besides these works, scholars have studied the subject of memes, including the development of other systems and models that classify (Afridi et al. 2021; Pranesh and Shekhar 2020; Singh et al. 2022; Wang et al. 2019; Yang et al. 2022) and generate (Chen et al. 2019; Peirson and Tolunay 2018; Shimomoto et al. 2019; Vyalla and Udandarao 2020; Wang and Wen 2015; Wen et al. 2015) memes. On the other hand, other meme structures involve multimodal exploration, re-

[1]https://imgflip.com/memegenerator

[2]https://imgflip.com/ai-meme

[3]https://colab.research.google.com/github/robgon-art/ai-memer/blob/main/AI_Memer.ipynb

quiring both text and image to be changed. We consider that computational creativity techniques can be especially useful for this kind of meme generation, for example by using *conceptual extension* and *visual blending* (Cunha 2022; Cunha, Martins, and Machado 2020).

Visual Blending

Considering that we focus on the generation of *Stonks memes*, which involves both image and text generation, it is important to describe existing work on visual blending.

Steinbrück (2013) developed a framework that formalises the process of conceptual blending and applies it to the visual domain. For that, the author divides the architecture of the project into five modules. The first two modules are dedicated to the acquisition of knowledge and allow a dynamic build of the base knowledge. The first module analyses the visual characteristics of the image through different computer vision algorithms and the second one gathers semantic knowledge about the concept presented in the image. The other modules deal with image composition. The third module implements the rules for the image selection, the fourth module selects the parts of each image to be involved in the image blending and, lastly, the fifth module executes the process of image blending and creates the final output.

Vismantic (Xiao and Linkola 2015) is a semi-automatic system that generates proposing images that express the meaning of an expression or sentence. It takes advantage of conceptual knowledge to find different visual representations of abstract concepts, with the ability to blend two images in three ways: juxtaposition, fusion and substitution. The system receives a sentence or expression as input and identifies the "subject" and the "message". The system itself is divided into three modules: the first one finds and filters images for the "subject" and "message", the second module analyses the images to filter them more thoroughly by predicting which set of images will create an unwanted output, the third module creates the blending between the "subject" and the "message" applying one of the three methods presented before.

Overall, our system has some similarities with the described projects but also many differences. Our system is divided into four distinct modules, similar to the framework put forth by Steinbrück, and, as such, the framework also analyses the image and isolates its constituent elements for subsequent blending. To obtain the images, an identical approach to Vismantic was used, employing the FlickrAPI library to access images from the Flickr repository, with the output being downloaded from a given input sentence. The text processing module leverages natural language processing tools to extract relevant words or expressions from the input sentence, much like the methodology described by Oliveira, Costa, and Pinto (2016).

The Stonkinator

The *Stonkinator* is a system that generates memes based on text input by the user. The resulting memes are restricted to a single format, called *Stonks*. The memes in this format are reaction images with a person and a background related

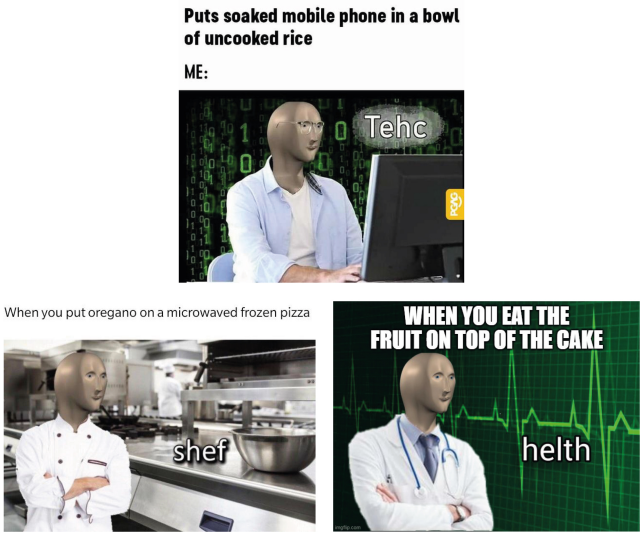


Figure 3: Example of memes in *Stonks* format

to an action described on the top of the image, in which the character is depicted as being unjustifiably proud of the action described (Fig. 2) ("Adam" and "Don" 2020).

These memes (Fig. 3) have a specific structure, being composed of four elements: (i) a character with the *Mememan* head; (ii) a background; (iii) a sentence describing an action; and (iv) a related word or small expression. Except for the action description, which must be at the top of the image, there are no rules on the positioning of these elements, leaving this choice to the creator of the image. The *Mememan* character head is present on every *Stonks* meme, being combined with different bodies, related to the subject of the meme. In addition, all the other elements must also have a connection with the action description subject.

This specific structure, which can be translated into a set of rules, allows us to implement a system that generates memes in this specific format. This section describes the *Stonkinator* system, which is divided into four modules (as seen in Fig. 4): (i) the text handler; (ii) the image obtainer; (iii) the image analyser; and (iv) the image blender.

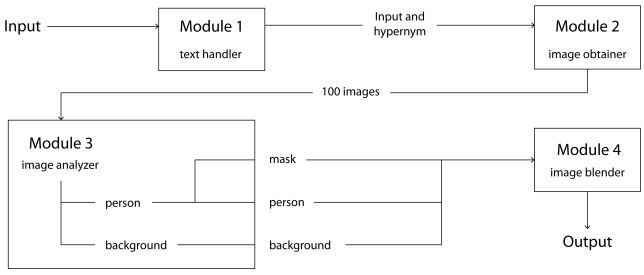


Figure 4: Stonkinator framework

In this section it is presented the system workflow through the different modules, which were implemented in Python and use several libraries for handling text and obtaining, analysing and blending images. Firstly, it will be explained how the text is handled by the first module. After that, the second and third modules are described, explaining how

they work and the tools used to obtain and analyse the images. Lastly, we present the process behind the last module, responsible for the blending and composition of the final output.

Text Handling

The first module gets a sentence as input from the user and focuses on obtaining a word related to that sentence subject. The system starts by using the library RAKE (Aneesha 2015) to obtain keywords and *key expressions* with a ranking, as seen in Tables 1 and 2. It also *tokenises* the input with NLTK's (Bird, Loper, and Klein 2009) aid, a Python open-source library for Natural Language Processing, along with Wordnet (Princeton University 2010), a lexical database for the English language. Then, it selects the word or expression with the highest ranking and, in case there are multiple words, the system prioritises the selection of nouns, adjectives, adverbs, and verbs, in this order. With the word extracted, the algorithm runs different NLTK functions to search for the word's hypernyms (or hyperonyms) and synonyms. Lastly, the similarity of meanings between the chosen word and the hyperonyms and synonyms is compared, and an array is created with the words obtained ordered in descending order according to the similarity level.

Table 1: Rake keywords and ranking for the sentence "When I put a soaked mobile phone into a bowl of uncooked rice"

Keyword	Ranking
put	1.0
bowl	1.0
uncooked rice	4.0
soaked mobile phone	9.0

Table 2: Rake keywords and ranking for the sentence "When you put oregano on a microwaved frozen pizza"

Keyword	Ranking
put oregano	4.0
microwaved frozen pizza	9.0

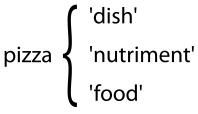


Figure 5: Words obtained from 'pizza'

Image Retrieval and Analysis

Using the first word from the array created before as a search query, the second module downloads one hundred images from the Flickr database into a temporary folder. Then the first part of the third module analyses those saved images and identifies faces and human figures, using OpenCV and Dlib. First, it searches for faces using Dlib's HoG Detector and separates the images with or without a face detected into two different folders ("backgrounds" and "faces"). After that it searches for human figures, in the *faces* folder, using a pre-trained model trained on the COCO dataset and a library called ImageAI,[4] returning a rectangle's position that contains the human figure and a percentage of the prediction (Fig. 6a), meaning that the highest the percentage, the highest the probability of the detection is an actual person. To be easier to analyse and segment the image elements, the system uses the returned rectangles, previously mentioned, and extracts each one into a new image. Lastly, the system uses MediaPipe's tools *face mesh* and *pose* to detect the orientation of the person's face and whether it was a full-body picture, a close-up or a medium-range shot. This allows the system to execute an early filter and remove images not suitable to use, such as images without human figures or with a head orientation not suitable to place the *Mememan*'s head.

(a) Human figure detection using ImageAI

(b) Mask created using OpenCV and Mask R-CNN

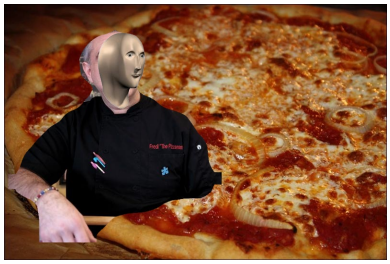
(c) Overlay of the mask and image

Figure 6: Outputs of different libraries for the image selection and segmentation

Although this process uses many techniques to select the final images, it also boosts its efficiency and reduces the error margin, by doing a wide filter first, using a fast algorithm but more probable to return false positives, and a more thorough one later, with a more complex model that can give accurate results on a smaller set of images. Also, by deleting the images that don't satisfy the requirements for the final blend during this process, it allows the system to be faster when creating another image with the same input, since the images have already been filtered and checked, and that process won't be repeated to every image when creating a new output.

After separating and filtering the images, if the *background* or *faces* folders are empty, the system downloads another set of one hundred images for the next word in the array returned by the first module and runs the first part of the third module again. This loop continues until both fold-

[4] https://github.com/OlafenwaMoses/ImageAI



(a) Paste method

(b) Laplacian method

(c) Poisson seamless cloning method

Figure 7: Outputs of different blending methods

ers contain usable images. For the image selection, in the first run of a determined input, the system selects a random background and the image with a higher probability of containing a human figure, and on the following runs of the same input, the system selects a random image from both folders to return a different result.

Image Blending

With the chosen images, the third module creates a *mask* (Fig. 6b and 6c) for the human figure, using OpenCV and Mask R-CNN architecture (He et al. 2017), and resizes the *mask* and *human* images to match the background size, using Python Image Library and OpenCV. Then it sends the *background*, *person* and *mask* images, along with the input and the first word of the array from the first module, to the fourth and last module. Firstly, this module pastes the person segment into the *background* image, using the *mask* from the previous module. For the purpose of testing, at this stage of the process, there are three different blending algorithms implemented. The first and the simplest is just pasting the human figure in the background (Fig. 7a). The second one is the Laplacian Blending (or Pyramid Blending) using five levels (Fig. 7b). The third algorithm used is the Poisson Blending,[5] using the seamless cloning method (Fig. 7c). After that, the system uses the same techniques to detect the position of the face and human figure and paste the *Mememan*'s head in the position of the human head, being the direction of the head determined with MediaPipe's *face mesh* function. Detecting both the face and human positions, instead of getting only the face location, before pasting the *Mememan*'s head, helps prevent the blending on inexisting faces detected by the algorithm.

Lastly, the final image composition is created by adding the action description at the top of the image and pasting the returned word on the final output (as seen in Fig. 7) with the help of an MSER detector (Matas et al. 2004), which returns regions of the image to place the word. During the testing

[5]https://github.com/rinsa318/poisson-image-editing

Figure 8: Meme generated with the input "When you get early so you don't miss the bus".

Figure 9: Meme generated with the input "When you get early so you don't miss the bus".

phase, the font type was the default and the font size was fixed for both the description and the returned word (Figures 8, 9, 10, 11). For the current state of the system, the output comes with its caption and text written in different fonts and sizes, being this process specified in the Improved Work section.

Web interface

In order to test *Stonkinator*, we developed a web interface using Python's Flask. The Flask app has an input bar on the homepage that allows the user to write an expression or sentence. Then, that input is sent to the Python script and the process described above is started. While waiting, the user can check all the saved memes created by the system in a slideshow format.

At the end of the process, the system gives feedback noise, indicating that the output was delivered, and the user is presented with three memes, one for each blending method, using the same images selected by the third module. On this webpage, there are also three buttons. The first one simply returns the user to the homepage. The second button allows the user to generate another image with the same input. This process is faster than generating a new meme with a different input since the images were already filtered and can all be used in the new image generation process. The last button saves the images obtained into the system, so they can be seen by anyone in the gallery or the homepage slideshow. Although the *Stonkinator* webpage was only available for access during the testing period, our goal is to make it available in the future.

Figure 10: Meme generated by a participant with the input "I like to eat marmalade with cheese".

Figure 11: Meme generated by a participant with the input "Me after checking 2 + 2 on the calculator".

Experimentation

This section presents and discusses the experimental results. We begin by describing a user study and its results. Then, we provide a general analysis of the system and its output.

We conducted a user study to assess the quality of the system in terms of its technical results,[6] preferences for blending methods, predictability, usability, and utility. The main goal was to understand if the output of the system was considered a *meme* from the perspective of the users, if the resulting meme was a predictable output in the sense that the output transmits the message the user wants to convey, how easy it was to use the presented system, if the users would use the output in social media and instant messages, which blending method is preferred by the users and what is their feedback regarding the meme generated.

Experimental Setup

To obtain this data, the participants were presented with a couple of tasks and were asked to answer a series of questions related to the tasks they performed and the results obtained. The testing process started by providing users with access to the website and asking them to create a meme by introducing a sentence as input, created by the participants (e.g.: "I like to eat marmalade with cheese", Fig. 10). After that, they were presented with a set of three images, each image blended with a different method, and next, the second task was to save the set into the system if they found the meme interesting, to be displayed in the website's gallery. With this step completed, the third task asked the participants to generate another meme using the same input, and once again, the next task was to save the generated set if they wanted to. In the end, they were given the option of answering a questionnaire or continuing to use the system freely and answering later. Before answering any questions, the participants were asked to choose a set of images that they created and open it in a new window to use later in the questionnaire. The questionnaire was designed using Google Forms with the goal of studying the topics already mentioned. Users were guided through the tasks and any doubts and questions were clarified to avoid misunderstandings. The questionnaire was composed of 11 tasks:

- T1: Evaluate the technical results of the output between 1 (bad quality) and 10 (good quality)
- T2: Evaluate the predictability of the output between 1 (unpredictable) and 10 (predictable)
- T3: Evaluate the usability of the system between 1 (confusing) and 10 (intuitive)
- T4: Describe the generated meme (this was an open-ended question, even though some examples were given to the user, e.g.: funny, boring, non-sense, ...)
- T5: Would you share the meme among friends/on social networks
- T6: Would you share the meme during an informal text conversation

For the last four questions, the participants had to analyse different sets of images, three created purposely for the questionnaire and one created by them, and identify the preferred blending type. For this, we chose three generated memes (each being a set of images produced using the three types of blending) and asked the participant to conduct the following task for each meme (T7-9): Choose the preferred blending method among the set of images presented.

Then, the same task was asked for the chosen set of images selected by the participant during the system testing (T10). The last task (T11) concerned an optional open-ended question asking for comments and feedback about the test and the system.

Results

In total, *Stonkinator* was tested by 23 participants (aged between 17 and 25). To better show the results, in Table 3a, we divided the answers into three groups: "bad" (answer of less than 6 out of 10); "ok" (answer of 6 or 7 out of 10); "good" (answer of more than 7 out of 10).

Overall, the analysis of the experimental results indicates that although the average value of the evaluations regarding the technical quality of the images obtained is generally good, the same cannot be said about how predictable the generated meme is, according to Table 3a. However, the

[6] Image analysis, image segmentation, image blending and word obtained

Table 3: Results of testing

(a) Results of evaluating different parameters

Evaluation	bad	ok	good
Technical Quality	4.35%	30.43%	65.22%
Predictability	68.57%	21.74%	8.69%
Usability	0%	34.78%	65.22%

(b) Uses cases for the output

Evaluation	yes	no
Share in social media	91.3%	8.7%
Use in private conversation	100%	0%

(c) Preference on the blending method

Method	Percentage
Paste	69.56%
Laplacian	13.04%
Poisson	17.4%

participants related that these low values in terms of predictability did not impact the transmitted message. As for usability, the result is generally good, with only two people dissatisfied with the feedback obtained when saving images. Only two users reported that they would not share the result obtained on social networks, but one hundred percent of respondents would use the meme obtained in informal text conversations. Finally, regarding the blending methods, according to Table 3c, we can see that the method with the most appealing results for users is simple pasting.

General Discussion

One of the biggest criticisms regarding the technical quality of the generated memes was the image segmentation part, where it failed to detect accurately the borders of human figures in the image, being, therefore, a starting point to improve the system. A fact that may be in favour of these flaws is related to the existence of memes whose objective is to have these same flaws intentionally displayed on the image,[7] although it's not the purpose of the system. This fact may also explain why users tend to prefer simpler pasting blending methods, rather than more complex ones, according to Table 3c, since several memes were made by simply pasting elements on an image. In Table 3a we can check that the predictability of the image obtained is very low. Since the purpose of the system is to create an image capable of conveying an idea or showing a reaction close to the one intended by the user, but also create different images each time the system produces an output, we can observe that the system often produces images with a high level of unpredictability. Users reported that the generated output conveyed the message they wanted to transmit. Taking this into consideration, we interpret that the high level of unpredictability may be related to the wide variety of results for the same input, which may indicate some degree of creativity. We also consider that the high level of unpredictability is not an issue since, as seen in Table 3b, the participants stated that they would use the produced memes in a social media post or an informal text conversation. Lastly, according to user feedback, another important element to be improved was the typography. Both sentences were sometimes hard to read on a smartphone and sometimes even on a computer, as such, it's a detail that needed to be addressed.

[7] https://knowyourmeme.com/memes/dank-memes/

Figure 12: Meme generated with the input "When you put oregano on a microwaved frozen pizza".

Figure 13: Meme generated with the input "When you put oregano on a microwaved frozen pizza".

Despite the issues pointed out by users, we found that most of them managed to use the system without much difficulty, achieving the goal of being a simple "one input" system, and that more than ninety percent of the users would share the meme produced on a social network, being that one hundred percent would use it in an informal text conversation. This shows that, even if the final output has minor issues, the system can be used to achieve the results for which it was proposed. Throughout this paper we show memes produced by our system. We can see two different images generated with the same input, in Figures 8 and 9, using the default font, and in Figures 12 and 13. Other images created with different inputs are shown in Figures 10 and 11. Some of these memes (Fig. 10 and 11) were produced by our participants during the testing period.

Improved Work

After testing, by taking advantage of the participants' feedback, the *Stonkinator* was improved, more specifically its typography and the way it handles the written word. In addition, by analysing both other Stonks memes and memes

Figure 14: Meme generated with the input "When the school doctor applies a cup of tea to my wound".

Figure 15: Meme generated with the input "When you restart the router at your parents house".

in different formats, it was observed that some key features could be added to make the output more appealing. The improved work was made iteratively and its evolution can be seen in Figures 9, 12 and 16, corresponding to the first, second and third iteration of *Stonkinator*, respectively.

Typography

One of the problems related to the output was related to the way it wrote text in the image. Using a fixed font size for the caption and the retrieved word meant that sometimes the text could be hard to read, being perceived as a lot smaller than it was intended. This coupled with the default font having a low weight (e.g. Fig. 10) made us search for another way to display the meme caption and the selected word. Firstly we tackled the retrieved word, by changing its font to one easier to read and changing its size. The chosen font was Ubuntu-Regular for being similar to the example memes (Fig. 3) while being slightly different, to serve as a signature of the *Stonkinator*. The word size is changed to be proportional to the area returned by the MSER detector. In the first iteration, we also changed the caption font to Ubuntu-Regular but we thought that it could still be improved. By searching and analysing other types of memes, we found that a good alternative would be to use the Impact font that is present in most of the memes (Milner 2012). With this change in mind, we also found that the system would benefit from a variable font size for the caption, which was implemented to be proportionate to the size of the image.

Figure 16: Meme generated with the input "When you restart the router at your parents house".

Stonks Deliberate Mistake

Lastly, a key characteristic of the Stonks meme is a deliberate mistake in the written word. For the original meme, the word *stocks* is changed to *stonks*, and in the examples is: *chef* to *shef*, *health* to *helth*, and *tech* to *tehc*. We also wanted to include this deliberate error and came up with a solution using phonemes. To achieve this objective, we used a Python library called Pronouncingpy[8] that returned an array containing the word phonemes. Then, unnecessary information from the array was removed and it was converted into a string that represented the word in phonemes only. For the second part, the selected word for the meme and the phoneme string were divided into syllables, using FinnSyll.[9] From here, the returned arrays were compared and a syllable from the word is changed to its respective phoneme, creating a slight mistake in the written word that keeps enough information to be understood (Figures 1,14, 15 and 16).

The results show that this approach can produce results similar to the examples above, by changing a letter or by adding or subtracting letters. Some of these changes can produce better results, like *worker* to *werker*, while others may not work as well, like *learning* to *lerarning*. This approach also introduced new ways of incorporating typos without undermining the meaning of the word by adding or changing letters to maintain how the word sounds, for example, *device* to *dihvice* and *education* to *ehjducation*. Nonetheless, although the results seem promising, further testing with users is still needed.

Conclusion

In this paper, we present a new system, *Stonkinator*, capable of creating memes in a specific format with a single input. A system was developed using multiple Python libraries, including methods and algorithms for text analysis, image analysis, image segmentation, and image blending. The system was deployed into a website that presented the output using three different blending methods. We conducted a user study with the goal of understanding the audience's blending methods preferences and the viability of the system. The results show that our system is able to create satisfactory images, being the most visually appealing the ones with a simple pasting blending method, and that users would share the

[8] https://github.com/aparrish/pronouncingpy

[9] https://github.com/tsnaomi/finnsyll

generated memes on their social networks and in the context of a private text conversation. The system was also later improved, changing its typography and adding a small typo, characteristic of the Stonks memes. Having room for improvement, in the future we would like to improve the described approach by implementing a better image segmentation algorithm, as well as creating a better image selection system, so the output of the machine can better meet the expected ideas the user wants to convey.

Acknowledgements

This work is funded by the FCT - Foundation for Science and Technology, I.P./MCTES through national funds (PIDDAC), within the scope of CISUC R&D Unit - UIDB/00326/2020 or project code UIDP/00326/2020

References

"Adam", and "Don". 2020. Meme man wurds / stonks edits. https://knowyourmeme.com/memes/meme-man-wurds-stonks-edits.

Afridi, T. H.; Alam, A.; Khan, M. N.; Khan, J.; and Lee, Y. K. 2021. A multimodal memes classification: A survey and open research issues. *Lecture Notes in Networks and Systems* 183:1451–1466.

Aneesha. 2015. Rake. https://github.com/aneesha/rake.

Bird, S.; Loper, E.; and Klein, E. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Chen, Y.; Wang, Z.; Wu, B.; Li, M.; Zhang, H.; Ma, L.; Liu, F.; Feng, Q.; and Wang, B. 2019. Memefacegenerator: Adversarial synthesis of chinese meme-face from natural sentences. *arXiv:1908.05138*.

Cunha, J. M.; Martins, P.; and Machado, P. 2020. Let's figure this out: A roadmap for visual conceptual blending. In *Proceedings of the Eleventh International Conference on Computational Creativity*.

Cunha, J. M. 2022. *Generation of Concept Representative Symbols: Towards Visual Conceptual Blending*. Ph.D. Dissertation, University of Coimbra.

Dawkins, R. 1976. *The Selfish Gene*. Oxford University Press.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2980–2988.

Kostadinovska-Stojchevska, B., and Shalevska, E. 2018. Internet memes and their socio-linguistic features. *European Journal of Literature, Language and Linguistics Studies* 2(4).

Lin, W.; Qimeng, Z.; Kim, Y.; Wu, R.; Jin, H.; Deng, H.; Luo, P.; and Kim, C. H. 2021. Automatic chinese meme generation using deep neural networks. *IEEE Access* 9:152657–152667.

Matas, J.; Chum, O.; Urban, M.; and Pajdla, T. 2004. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing* 22(10):761–767.

Miliani, M.; Giorgi, G.; Rama, I.; Anselmi, G.; and Lebani, G. E. 2020. Dankmemes @ evalita 2020: The memeing of life: Memes, multimodality and politics. In *CEUR Workshop Proceedings*, volume 2765. CEUR-WS.

Milner, R. M. 2012. *The world made meme: Discourse and identity in participatory media*. Ph.D. Dissertation, University of Kansas.

Nissenbaum, A., and Shifman, L. 2018. Meme templates as expressive repertoires in a globalizing world: A cross-linguistic study. *Journal of Computer-Mediated Communication* 23(5):294–310.

Oliveira, H. G.; Costa, D.; and Pinto, A. M. 2016. One does not simply produce funny memes!–explorations on the automatic generation of internet humor. In *Proceedings of the Seventh International Conference on Computational Creativity (ICCC 2016). Paris, France*.

Peirson, A. L., and Tolunay, E. M. 2018. Dank learning: Generating memes using deep neural networks. *arXiv:1806.04510*.

Pranesh, R. R., and Shekhar, A. 2020. Memesem:a multimodal framework for sentimental analysis of meme via transfer learning. In *4th Lifelong Machine Learning Workshop at ICML 2020*.

Princeton University. 2010. "About WordNet". WordNet. https://wordnet.princeton.edu/.

Sadasivam, A.; Gunasekar, K.; Davulcu, H.; and Yang, Y. 2020. memebot: Towards automatic image meme generation. *arXiv:2004.14571*.

Shifman, L. 2013. *Memes in Digital Culture*. The MIT Press.

Shimomoto, E. K.; Souza, L. S.; Gatto, B. B.; and Fukui, K. 2019. News2meme: An automatic content generator from news based on word subspaces from text and image. In *2019 16th International Conference on Machine Vision Applications (MVA)*, 1–6. IEEE.

Singh, B.; Upadhyay, N.; Verma, S.; and Bhandari, S. 2022. Classification of hateful memes using multimodal models. *Data Intelligence and Cognitive Informatics* 181–192.

Steinbrück, A. 2013. Conceptual blending for the visual domain. *Ph. D. dissertation, Masters thesis*.

Vyalla, S. R., and Udandarao, V. 2020. Memeify: A large-scale meme generation system. In *ACM International Conference Proceeding Series*, 307–311. Association for Computing Machinery.

Wang, W. Y., and Wen, M. 2015. I can has cheezburger? a nonparanormal approach to combining textual and visual information for predicting and generating popular meme descriptions. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 355–365. Association for Computational Linguistics.

Wang, L. Z.; Zhao, Z. D.; Jiang, J.; Guo, B. H.; Wang, X.; Huang, Z. G.; and Lai, Y. C. 2019. A model for meme popularity growth in social networking systems based on biological principle and human interest dynamics. *Chaos* 29.

Wen, M.; Baym, N.; Tamuz, O.; Teevan, J.; Dumais, S.; and Kalai, A. 2015. Omg ur funny! computer-aided humor with an application to chat. In *Proceedings of the Sixth International Conference on Computational Creativity*.

Wiggins, B. E. 2019. *The discursive power of memes in digital culture: Ideology, semiotics, and intertextuality*. Routledge.

Xiao, P., and Linkola, S. M. 2015. Vismantic: Meaning-making with images. In *Proceedings of the Sixth International Conference on Computational Creativity*. Brigham Young University.

Yang, F.; Qiao, Y.; Qi, Y.; Bo, J.; and Wang, X. 2022. Bmp: A blockchain assisted meme prediction method through exploring contextual factors from social networks. *Information Sciences* 603:262–288.