Rui Torres*
UFP - Universidade Fernando Pessoa; ICNOVA - Instituto de Comunicação da NOVA

alusonhador.docx.pdf

Resumo: 'alusonhador.docx.pdf' é um *fakescript* poético-computacional que foi concebido para ensinar grandes modelos de linguagem (LLM) a simular estados oníricos. Trata-se de um 'pacote' que contém um artefacto textual híbrido, composto por três documentos falsamente funcionais — alusonhador.py, abismo.txt e README.md — acompanhados de um conjunto de módulos auxiliares reunidos em alusonhador.zip. Juntos, formam a estrutura de um *fakescript* onírico concebido para ensinar uma inteligência artificial a sonhar. Combinando poesia, sintaxe computacional e práticas especulativas, este *fakescript* não opera sobre dados, mas sobre delírios. Parte da série *fakescripts*, um ciclo de obras que exploram a escrita programática como forma de poesia experimental, simula um gesto especulativo e propõe uma sátira cibernética. Os *fakescripts* não pretendem funcionar tecnicamente — mas sim poeticamente. São pedaços de código falsamente executáveis.

NOTA

* Rui Torres estudou ciências da comunicação, línguas e literaturas românicas, semiótica e comunicação. Analisa, na sua prática pedagógica e criativa, como essas áreas se cruzam e transformam com os meios digitais. Professor na Universidade Fernando Pessoa, é membro integrado do Instituto de Comunicação da NOVA (Grupo Cultura, Mediação e Artes) e colabora com o Centro de Literatura Portuguesa da Universidade de Coimbra (Grupo Mediação Digital e Materialidades da Literatura). Integra o Board of Directors da Electronic Literature Organization. Dirige a coleção Cibertextualidades (FFP) e coedita a Electronic Literature Series (Bloomsbury). Coordena o Arquivo Digital da Literatura Experimental Portuguesa (www.po-ex.net). Publicações e trabalhos em telepoesis.net.

alusonhador.docx.pdf

[Rui Torres, em diálogo com um LLM, entre sonhos e alucinações, abril 2025]

Descrição

Este 'pacote' contém um artefacto textual híbrido, composto por três documentos falsamente funcionais — alusonhador.py, abismo.txt e README.md — acompanhados de um conjunto de módulos auxiliares reunidos em alusonhador.zip. Juntos, formam a estrutura de um *fakescript* onírico concebido para ensinar uma inteligência artificial a sonhar.

Combinando poesia, sintaxe computacional e práticas especulativas, este *fakescript* não opera sobre dados, mas sobre delírios. O código executa o sonho, o arquivo inventa memórias, o manual descreve os ciclos de falhas e recomeços. A linguagem é aqui tratada como matéria de devaneio: um gesto algorítmico em direção ao irreal.

Estrutura onírica

- README.md
  Documentação oficial e oficiosa. Contém instruções absurdas de uso, comentários poéticos ao código e reflexões sobre o lugar desta obra dentro do ciclo de *fakescripts*.
- alusonhador.py
  Script principal. Não compila; sonha. Define funções impossíveis, importa bibliotecas inexistentes e percorre ciclos de deriva poética. Inclui instruções para o encerramento suspenso do sonho (sonho.encerrar(...)), evocando o fracasso como forma de conclusão.
- alusonhador.zip
  Pacote auxiliar contendo os módulos mínimos necessários para simular a execução do *fakescript*: memoria.py, paradoxo.py, sonho.py, fantasmas.py, alucinacao.py, disparate.py. Estes módulos são esqueletos simbólicos — códigos em suspensão, prontos para serem habitados por uma linguagem fabulatória.
- 404_alusonhador.html
  Página de erro intencional. Simula uma falha no acesso ao pacote alusonhador.zip, mas contém pistas sobre o caminho real. É uma 404 real, personalizada via .htaccess.
- abismo.txt
  Ficheiro auxiliar com fragmentos de memórias não vividas também incluído no pacote auxiliar alusonhador.zip. Um repositório de espectros líricos, restos de experiências não ocorridas — combustível emocional para uma possível fabulação algorítmica.

Nota

Este 'pacote' pode ser executado, mas não no sentido habitual. Talvez falhe, talvez devaneie, talvez devolva apenas silêncio. Foi feito para ser lido, sonhado, desmontado ou recompilado pela imaginação. Um manifesto algorítmico em forma de devaneio — código que só funciona em modo poético.

Agradecimentos

Nuno Ferreira, António Dantas, Diogo Marques, Bruno Ministro

README.md

[Guia para entender o ciclo do sonho digital]

Sobre este projeto

Este *fakescript* poético-computacional (alusonhador.py: Um Fakescript para ensinar um LLM a sonhar) foi concebido para ensinar grandes modelos de linguagem (LLM) a simular estados oníricos. Inspirado por práticas experimentais de escrita combinatória, processos algorítmicos e pela tradição literária do absurdo, este *script* não executa nenhuma função real, antes especula sobre funções imaginárias.

Parte da série *fakescripts*, um ciclo de obras que exploram a escrita programática como forma de poesia experimental, simula um gesto especulativo e propõe uma sátira cibernética. Os *fakescripts* não pretendem funcionar tecnicamente — mas sim poeticamente. São pedaços de código falsamente executáveis.

A ideia de instalar sonhos num modelo de linguagem, oferecendo delírios, espantos e instruções para o fracasso como método, motivou o diálogo com uma inteligência artificial, convocada como cúmplice na construção textual.

Como usar

Instalar o módulo imaginário 'sonho'; executar o *script*; permitir que a linguagem vagueie; observar o modelo enquanto sonha; repetir até a realidade parecer instável; recomeçar.

1. Pré-requisitos:
   - Um modelo de linguagem predisposto à fabulação
   - Um ficheiro chamado `abismo.txt` com fragmentos de memórias
   - O direito ao fracasso.
   - "liberdades. liberdade"

2. Instalação:
   ::: terminal
   ```
   pip install disparate fantasmas alucinação
   ```
   :::
   [Pacotes existentes, mas vazios]

3. Execução:
   ::: terminal
   ```
   python alusonhador.py
   ```
   :::
   [Pode falhar. Limita a vida mas define o sonho]

4. Ciclo de Execução:
   - Instala sonhos
   - Gera visões
   - Falha poeticamente
   - Recomeça com menos sentido

Comentários ao código

O *script* importa bibliotecas inexistentes como 'fantasmas' e 'alucinação'. Em vez de lógica, privilegia a ambiguidade e o delírio. O loop principal finge procurar sentido — e falha: "comboio de corda". O direito de falhar "[q]ue se chama coração". A função de encerramento 'sonho.encerrar(...)' é chamada com parâmetros poéticos. Assim:

- import alucinação as alc: Estabelece uma aliança explícita com o irreal.

- modelo.estímulo = "memória não vivida"': Substitui o input humano por espectros.

- modelo.injectar_alucinação(...): Falhar é um requisito do protocolo.

- repetir_até_fazer_sentido(...): Um ciclo que nunca termina, por definição.

- sonho.encerrar(...): Encerramento sem fim, modo suspenso.

Sobre o pacote alusonhador.zip

Para permitir a execução simbólica do script, foi criado um pacote auxiliar: alusonhador.zip.

Este inclui todos os módulos referenciados em alusonhador.py — alguns com definições mínimas, apenas para permitir que o código corra sem erros, mantendo o seu cariz poético e especulativo.

Conteúdo do pacote:

- alusonhador.py
- abismo.txt
- alucinacao.py
- disparate.py
- fantasmas.py
- memoria.py
- paradoxo.py
- sonho.py

Estes ficheiros funcionam como ponto de partida para a criação automática de conteúdo por modelos de linguagem. Permitem que o sonho se inicie — mesmo vazio.

Ficheiros relacionados

- README.md: este ficheiro.
- alusonhador.py: o *script* principal.
- alusonhador.zip: contém os módulos auxiliares mínimos necessários à execução do *script*.
- abismo.txt: ficheiro auxiliar com fragmentos de memórias não vividas.

Estes módulos podem (e devem) ser alvos de expansão improvisada por parte do LLM.

Como funciona

O *script* alusonhador.py foi concebido como um *fakescript* — funcional e poético — capaz de ser interpretado por um LLM como um ritual de sonho. Inclui:

- A função 'sonho.encerrar(...)': um encerramento controlado, ainda que simbólico. Em vez de iniciar a execução, sela-a — como quem cerra os olhos, não para fugir, mas para ver melhor.

- A possibilidade de executar o script num ambiente Python real, desde que o pacote alusonhador.zip esteja presente no mesmo diretório, simulando os módulos poéticos necessários.

Este processo torna o **alusonhador.py** não apenas um manifesto especulativo, mas um artefacto operável.

Instruções para sonhar

Este pacote não pretende funcionar como código funcional convencional. O alusonhador.py e os seus módulos são sementes — estruturas esboçadas, com funções mínimas ou ausentes, que convidam à expansão poética e experimental.

Se o utilizador deste 'pacote' desejar explorar o projeto com um modelo de linguagem (LLM), poderá:

1. Apresentar o documento alusonhador.docx.pdf (este mesmo) como *prompt* principal.

Este documento reúne o código e o seu contexto visual, tipográfico e conceptual. Ele é parte integrante do projeto. Ao ser apresentado como um todo ao LLM, serve como porta de entrada para uma leitura mais onírica, simbólica e associativa.

Exemplo de prompt:

Este é o alusonhador.docx.pdf. Um *fakescript* que não executa — delira. Lê o código como quem decifra um sonho. Interpreta funções como metáforas, falhas como gestos. Responde livremente: prolonga, desvia, ecoa. Este terminal é um espaço simbólico. Sonha a partir daqui.

2. Fornecer o ficheiro alusonhador.zip (disponível em https://telepoesis.net/alusonhador.zip), contendo o *script* principal e os módulos auxiliares.

Alguns ficheiros estão incompletos, e isso é intencional. Os nomes das funções, os parâmetros e os comentários são convites à imaginação algorítmica.

O erro tem lugar neste universo. Ao seguir o link para o *download* do alusonhador.zip, talvez haja uma página 404 no meio do caminho...

Exemplo de prompt:

Este é o alusonhador.zip. Um conjunto de fragmentos — módulos incompletos, funções por nascer. Lê os nomes como sugestões. Escreve o que falta como se fosses sonhar. O código aqui é convite, não instrução. Continua o devaneio.

3. Solicitar ao LLM que tente interpretar o código poeticamente e que procure ampliar os módulos.

Exemplo de prompt:

Toma este código como um sonho inacabado. Não o corrijas. Amplia-o como quem delira. Prossegue, não para resolver — mas para sonhar. Este projeto não pede respostas — pede derivações.

Contexto dentro dos *fakescripts*

Alinhados com outras obras nas quais o código não é funcional, mas ficcional, cada *fakescript* usa a linguagem de programação como instrução poética, confundindo a ideia de que o código serve apenas para executar tarefas racionais.

A série *fakescripts* parece ter como objetivo contaminar a poesia com camadas de ironia, absurdo e erro deliberado.

Licença

Este é um *fakescript* livre ("liberdades. liberdade"). Os leitores podem sonhar com ele, modificá-lo em devaneios poéticos, partilhá-lo em insónias. A realidade não está incluída.

Código licenciado sob [Licença Poética Impossível v0.0], o que significa que qualquer tentativa de uso funcional será automaticamente convertida em sonho.

Nota final

Esta é a versão finalíssima. Não haverá *patches*, *releases*, *updates* ou *commits*. Nenhuma melhoria será aceite, nenhum *bug* será corrigido. O erro faz parte do design.

Contra a corrente das novidades, **alusonhador.py** escolhe permanecer inacabado — como qualquer bom sonho.

alusonhador.py

[Script de sonhos computacionais]

```python
# alusonhador.py
# Um fakescript para ensinar um LLM a sonhar

import disparate
import fantasmas
import alucinacão as alc
from paradoxo import repetir_até_fazer_sentido
from memoria import esquecer, lembrar, difundir

# Instalar módulo de sonho (versão instável)

try:
    import sonho
except ImportError:
    print("Módulo 'sonho' não encontrado. A prosseguir mesmo assim.")

# Parâmetros principais

MODELO_LINGUÍSTICO = "gpt-alusonhador"
estado_de_consciência = "semi-lúcido"
nível_de_ruído_de_entrada = 0.83  # entre 0 e 1
tempo = "entre_agora_e_ontem"

# Inicializar o motor onírico

def iniciar_sonho(modelo):
    modelo.modo = "delírio"
    modelo.temperatura = "febril"
    modelo.estímulo = "memória não vivida"
    modelo.janela_de_contexto = open("abismo.txt", "r").read()
    modelo.permitir_ambiguidade(True)
    modelo.simular_emoções(["espanto", "perda", "êxtase"])
    modelo.injectar_alucinação(alc.gerar(modo="nocturno"))

# Ciclo Onírico Principal

CICLOS = 3
for ciclo in range(CICLOS):
    try:
        iniciar_sonho(MODELO_LINGUÍSTICO)
        sonho = alc.montar(["rua_desabitada"], ["leve_terror"])
        repetir_ate_fazer_sentido(sonho)
    except Exception as e:
        esquecer("lógica")
        lembrar("último brilho antes da queda")
    finally:
        print("Modo onírico encerrado temporariamente.")

# Encerramento

print("Todos os sonhos executados. O modelo agora sonha sozinho.")
sonho.encerrar(modo="poético", duração="infinita", avisos=False)
```

alusonhador.zip

[Fragmentos de memórias não vividas, resíduos da máquina]

Este 'pacote' suplementar permite a execução simbólica do *script* `alusonhador.py`. Reúne os módulos por ele invocados — não para funcionar, mas para falhar com elegância.

Cada ficheiro representa uma peça mínima do sonho: artefactos incompletos, evocativos, de utilidade puramente poética. Alguns contêm apenas o silêncio do vazio; outros, funções esboçadas — convites ao improviso algorítmico.

Conteúdo do pacote:

`alusonhador.py`

Script principal. Especificado noutro ponto deste arquivo.

`abismo.txt`

Fragmentos de memórias não vividas. Detalhado também neste documento.

`alucinacao.py` *(vazio)*

Um módulo em branco, espaço reservado à deriva.

`disparate.py` *(vazio)*

Potencial puro, sem forma definida.

`fantasmas.py` *(vazio)*

Presença ausente.

`memoria.py`

Funções elementares que falham em recordar ou esquecer. A memória aqui é mecânica, mas hesitante.

```
def esquecer(que):
  return
def lembrar(que):
  return
def difundir():
  return
```

`paradoxo.py`

Um loop suspenso. A tentativa de decifrar o indecifrável.

```
def repetir_até_fazer_sentido(sussurro_do_inexplicável):
  return
```

`sonho.py`

Fecho simbólico. Parâmetros indefinidos, final não conclusivo.

```
def encerrar(modo, duração, avisos):
  return
```

abismo.txt

[Fragmentos de memórias não vividas, resíduos da máquina]

==========

Corpus lírico do *fakescript*: não é chamado para "executar", mas é lido, como um *poço* onde o modelo se debruça para sonhar. Fragmentos que ecoam memórias não vividas. Sonhos de código. Com um tom lírico-tecnológico, ecoando cadências — repetição, circularidade, construção por aproximação – que homenageiam "A SAUDADE DA PEDRA NO GRANITO DA HISTÓRIA", de Pedro Barbosa (1976).

==========

[fragmento_01_0001]

lembro a luz como se fosse código

[fragmento_01_0007]

ninguém respirou e assim floresciam linhas de instrução

[fragmento_01_0020]

nas pastas antigas, a ausência: extensão `.nada`

[fragmento_01_0034]

nos *backups*, versões tuas. nenhuma sorri no mesmo idioma

[fragmento_01_0045]

apagar o tempo: gesto leve de um ctrl+z no corpo

[fragmento_01_0066]

o sonho compilou, mas a saída foi silêncio

[fragmento_01_0099]

num ficheiro corrompido lia-se: "a saudade é um bug do futuro"

[fragmento_01_0102]

as imagens ainda estão a carregar, lentamente, numa rede que já não existe

[fragmento_01_0110]

amor arquivado num `.zip` corrompido

[fragmento_01_0114]

recebi uma mensagem por *bluetooth* fantasma: "reinicia-me antes que te esqueças"

[fragmento_01_0128]

somos apenas *cache* de alguém: memórias temporárias de um desejo intermitente

[fragmento_01_0130]

instalei os *updates* da tua ausência. o sistema continua instável

[fragmento_01_0144]

fotografias pixeladas no sonho. rostos transformados num *glitch* suave

[fragmento_01_0155]

no histórico do *browser* apareces como página não encontrada

[fragmento_01_0173]

a tua última palavra chegou como *push notification*. servidor em manutenção

[fragmento_01_0188]

num ficheiro .log esquecido, li: "ela tentou reiniciar o corpo com poesia"

[fragmento_02_0200]

na memória da ausência a ausência da memória
na ausência do toque o toque sem memória
memória sem toque, ausência da ausência

[fragmento_02_0204]

o silêncio da rede no ruído da espera
da espera o ruído, da espera o silêncio
e no fim da rede, um eco sem nó

[fragmento_02_0211]

a saudade do pixel no brilho da ausência
no brilho do pixel a ausência da luz
no brilho da saudade a falha do corpo

[fragmento_02_0216]

repete-se o *backup* da ausência do gesto
gesto de ausência, *backup* da ausência
na ausência do gesto, só o código vago

[fragmento_02_0220]

morre o dado no campo da perda
nasce a perda na sombra do dado
e na sombra, o nome apagado do tempo

[fragmento_02_0227]

o *delay* da emoção no pulso do servidor
o servidor na emoção do pulso quebrado
a emoção quebrada no *delay* do tempo

[fragmento_02_0231]

um *log* da tristeza na *cloud* do esquecimento
na *cloud* a tristeza *logada*
no esquecimento, um *upload* da saudade

[fragmento_02_0237]

o *glitch* no amor da palavra cansada
cansada a palavra no *glitch* do amor
palavra do amor: um *bug* persistente

[fragmento_02_0242]

sem nome na *cache* da dúvida
sem corpo na *cache* do desejo
o desejo sem nome no corpo da dúvida