

# **Inteligencia artificial ¿Qué es?**

- Máquinas.
- Informática.
- Software.
- Aprendizaje automático (Machine learning).
- Algoritmos.

*“La inteligencia artificial (IA) es una tecnología con **capacidades de resolución de problemas** similares a las de las personas. La IA en acción parece **simular la inteligencia humana**: puede reconocer imágenes, escribir poemas y hacer predicciones basadas en datos”* (Amazon Web Services, s.f.).

Entonces la inteligencia artificial es cuando una máquina o computador hace cosas que normalmente haría una persona usando obviamente su inteligencia.

## **Machine learning**

Rama de la inteligencia artificial. Se centra en que las computadoras y máquinas logren imitar la forma en que aprenden los humanos, en otras palabras estas máquinas aprenden solas en base a datos en vez de que sean programadas paso a paso.

Existen diferentes tipos de aprendizaje de machine learning, en esta ocasión se centrará en la investigación de los dos que se presentan a continuación:

### **1. Aprendizaje supervisado:**

*“Is defined by its use of labeled datasets to train algorithms to classify data or predict outcomes accurately”*(IBM, 2021).

Es una técnica de aprendizaje automático, donde se entrena usando un modelo de datos que ya tienen respuestas correctas, en otras palabras tiene parámetros de entrada y salida etiquetados por humanos.

Un ejemplo de esto podría ser: Si se quiere que un cierto modelo logre identificar entre perros y gatos, para esto se le entregan imágenes de perros y gatos claramente identificadas y/o clasificadas. De esta forma la máquina logrará identificar diferencias y similitudes entre ambos, para así con el tiempo lograr identificar por sí solo si una imagen corresponde a un perro o gato.

## 2. Aprendizaje generativo:

Es un tipo de aprendizaje en donde no solo se analizan datos, sino que también aprenden a crear datos nuevos y realistas, que serán parecidos a lo que vieron en su entrenamiento.

*“Implica el desarrollo de modelos que aprenden la distribución subyacente de los datos de entrenamiento. Estos modelos son capaces de generar nuevas muestras de datos con características similares al conjunto de datos original”* (GeeksforGeeks, 2024).

Aprendizaje	Supervisado	Generativo
Objetivo	<b>Predecir</b> una salida a partir de datos clasificados.	<b>Generar</b> nuevos datos similares a los originales.
Tipo de datos	Datos clasificados (entrada + salida).	Datos sin necesidad de etiquetas.
Ejemplo	<b>Clasificar</b> imágenes de gatos y perros.	<b>Crear</b> una imagen nueva de un gato.

# Instalación de IBEX

Programa o librería hecho en C++, sirve para resolver problemas matemáticos difíciles.

## ¿Qué relación podría tener ibex con la IA?

Se estudiará y generará una herramienta generativa para data sintética en el contexto de optimización global.

IBEx en esta investigación será el programa que resolverá para ver los resultados de los problemas de optimización que se creen en un modelo generativo.

Para esta instalación lo primero que se necesita es poder tener las siguientes aplicaciones en el sistema:

- g++
- doblar
- bisonte
- cmake
- git
- pkg-config

## Paso 1 : Instalación de herramientas necesarias.

Se utilizará la distribución ubuntu del sistema operativo linux.

```
valentina@Valentina-Lenovo:~$ sudo apt-get install -y g++ gcc flex bison cmake git pkg-config
```

Con esto instalamos las múltiples herramientas mencionadas antes que son de desarrollo esenciales en nuestro sistema.

## Paso 2 : Clonar el repositorio de ibex desde el github.

Se descarga el código fuente de ibex desde el github proporcionado por el profesor y se ingresa a la carpeta del proyecto.

```
valentina@Valentina-Lenovo:~$ git clone https://github.com/ibex-team/ibex-lib.git
cd ibex-lib
Clonando en 'ibex-lib'...
remote: Enumerating objects: 37535, done.
remote: Counting objects: 100% (537/537), done.
remote: Compressing objects: 100% (189/189), done.
remote: Total 37535 (delta 415), reused 349 (delta 348), pack-reused 36998 (from 2)
Recibiendo objetos: 100% (37535/37535), 91.80 MiB | 10.27 MiB/s, listo.
Resolviendo deltas: 100% (27241/27241), listo.
```

### Paso 3 : Crear una carpeta para compilar.

Se crea una carpeta build dentro del proyecto, que está destinada a contener todos los archivos generados durante la compilación del proyecto.

```
valentina@Valentina-Lenovo:~/ibex-lib$ mkdir build
cd build
```

### Paso 4 : Configurar la compilación con cmake.

Se ejecuta el cmake para poder configurar el proyecto antes de compilarlo, permitiendo así usar soplex como solver si está disponible.

```
valentina@Valentina-Lenovo:~/ibex-lib/build$ cmake -DLP_LIB=soplex ..
```

### Paso 5 : Compilar ibex.

Se compila el código de ibex con make, usando todos los núcleos disponibles para poder agilizar el proceso.

```
valentina@Valentina-Lenovo:~/ibex-lib/build$ make
```

### Paso 6 : Instalación de ibex en el sistema.

Se instala ibex en el sistema, esto permite acceder a sus ejecutables desde cualquier ruta del sistema.

```
valentina@Valentina-Lenovo:~/ibex-lib/build$ sudo make install
```

### Paso 7 : Verificación que esté todo bien.

Se verifica la instalación, esto devuelve la versión instalada de ibex.

```
valentina@Valentina-Lenovo:~/ibex-lib/build$ ibexsolve -v
IbexSolve Release 2.9.1.0
```

La documentación también incluye instrucciones para instalación en Windows, mediante Visual Studio y Chocolatey. Sin embargo, como mi sistema operativo es Ubuntu, seguí el proceso de instalación estándar para Linux utilizando cmake, make y make install.

## Instalación de GPT4ALL (modelo generativo).

GPT4All es una herramienta que permite usar modelos de lenguaje con una interfaz gráfica o bien de la misma terminal.

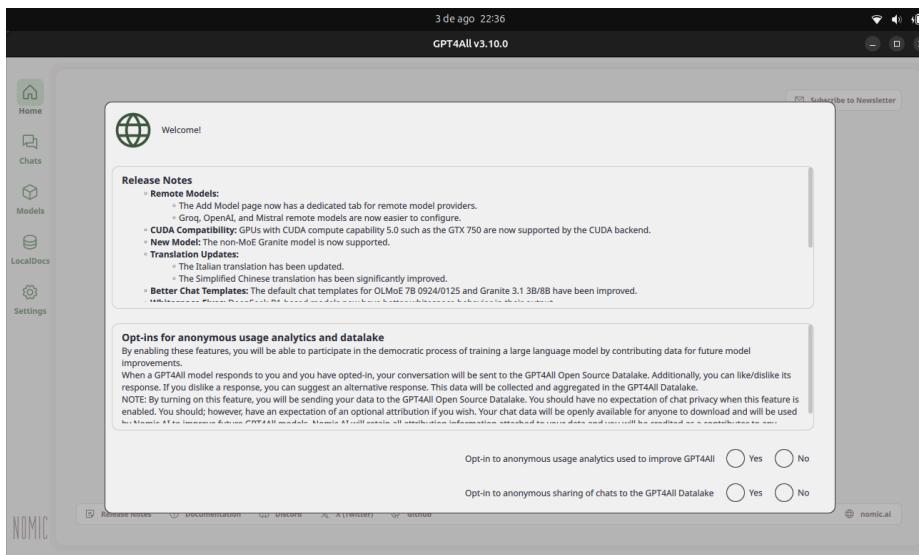
### Paso 1 : Descargar GPT4All.

Se utiliza el siguiente enlace para descargar "<https://gpt4all.io/index.html>" una vez descargado, abrimos terminal para darle permisos e instalar.

```
valentina@Valentina-Lenovo:~$ mkdir -p ~/Escritorio
chmod u+rwx ~/Escritorio
valentina@Valentina-Lenovo:~$ mkdir -p ~/MisProgramas/GPT4All
valentina@Valentina-Lenovo:~$ ~/Descargas/gpt4all-installer-linux.run
```

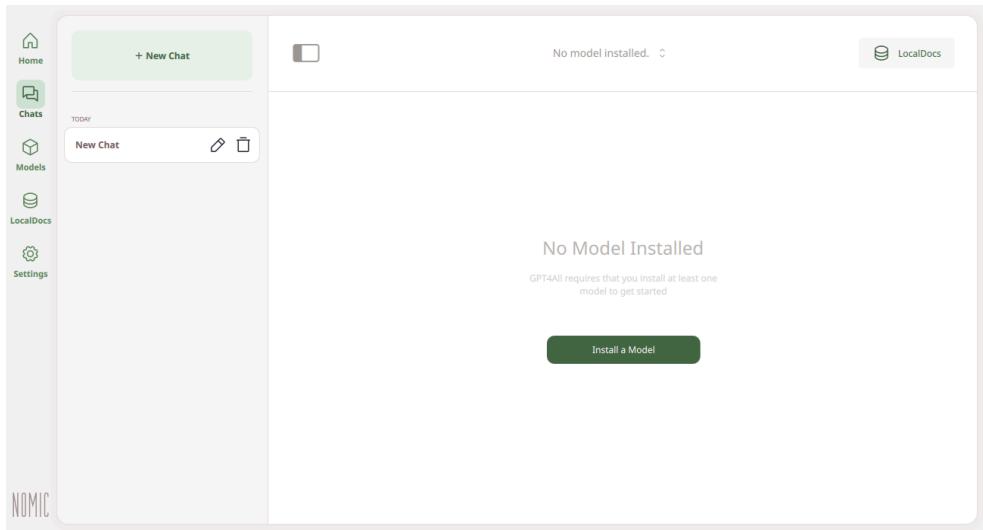
### Paso 2 : Verificar.

Se verifica que la instalación fue de manera correcta abriendo la aplicación.



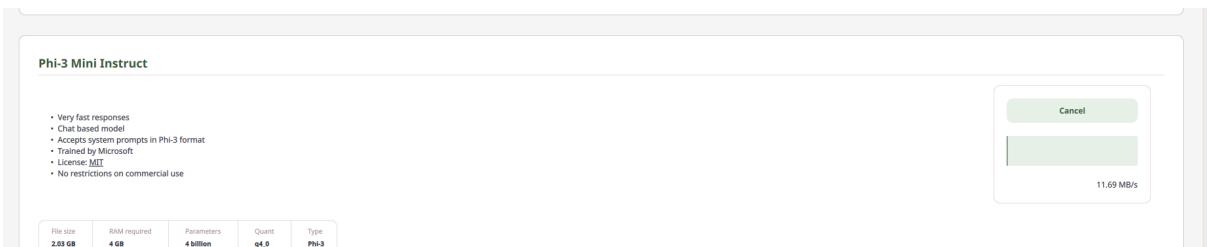
## Funcionamiento de GPT4All.

Como bien se dijo anteriormente se instaló GPT4All, lo cual es una aplicación de código abierto que permite utilizar modelos de lenguaje tipo LLM (Large Language Models (modelo extenso del lenguaje) de forma local y sin conexión a internet.



Como se ve en la imagen no hay ningún modelo descargado ya que GPT4All es solo el “contenedor” o entorno de ejecución, para que pueda generar texto y realizar tareas necesita cargar un modelo de lenguaje entrenado. En este caso se seleccionó el modelo **Phi-3 Mini Instruct** por su compatibilidad con

equipos de bajo recurso (requiere sólo 4 GB de RAM) su tamaño reducido (2GB) y su buen rendimiento en tareas de generación de lenguaje.



## Ejecución de problemas de optimización con IBEX.

### 1. Ubicación de los archivos de prueba:

Después de la instalación de la librería ibex, dentro de su estructura de carpetas se encuentran una serie de problemas de optimización que permiten poner a prueba el motor ibexopt.

Estos archivos se encuentran en:

- medium/
- blowup/
- hard/
- easy/
- others/
- unsolved/

```
valentina@Valentina-Lenovo:~$ ls
chat-unix-test Documentos gpt4all  Imágenes      Música      Público  Videos
Descargas   Escritorio ibex-lib MisProgramas Plantillas snap
valentina@Valentina-Lenovo:~$ cd ibex-lib/
valentina@Valentina-Lenovo:~/ibex-lib$ ls
appveyor.yml    build      cmake.utils  doc       examples  interval_lib_wrapper  lp_lib_wrapper  src
benchs          CMakeLists.txt  COPYING.LESSER  Doxyfile  ibex.pc.in  LICENSE           README.md      tests
valentina@Valentina-Lenovo:~/ibex-lib$ cd benchs/
valentina@Valentina-Lenovo:~/ibex-lib/benchs$ ls
optim  scatter_plot.gnuplot  solver
valentina@Valentina-Lenovo:~/ibex-lib/benchs$ cd optim/medium
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim/medium$ ls
alkyl.bch     ex14_2_7.bch  ex2_1_9fact.bch  ex6_1_4.bch  ex6_2_6.bch  ex7_3_4bis.bch  haverly.bch  like.bch
alkylbis.bch  ex2_1_7.bch  ex5_2_case3.bch  ex6_2_11.bch  ex6_2_8.bch  ex7_3_5bis.bch  himmel16.bch  schwefel5-abs.bch
bearing.bch   ex2_1_8.bch  ex5_3_2.bch   ex6_2_12.bch  ex6_2_9.bch  ex8_4_4bis.bch  launch.bch  schwefel5.bch
ex14_2_3.bch  ex2_1_9.bch  ex6_1_3bis.bch  ex6_2_14.bch  ex7_3_4.bch  ex8_5_2_1.bch  like-1.bch
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim/medium$
```

```
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim/blowup$ ls
ex2_1_7.bch    ex6_2_13.bch  ex7_3_5.bch    ex8_4_4.bch  ex8_4_6bis.bch  ex8_5_1.bch  hhfair1.bch  meanvar.bch  ramseybis.bch
ex2_1_9bis.bch ex7_2_4.bch  ex8_2_1.bch    ex8_4_5.bch  ex8_4_6ter.bch  ex8_5_2.bch  hhfair.bch   process.bch
ex6_1_1.bch   ex7_2_8.bch  ex8_4_4-1.bch  ex8_4_6.bch  ex8_5_1-1.bch  ex8_5_6.bch  hydro.bch   ramsey.bch
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim/blowup$
```

```
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim/hard$ ls
chem-1.bch chem.bch chembis.bch ex6_1_3.bch ex6_2_10.bch ex8_5_1bis.bch sambal.bch
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim/hard$
```

```
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim/easy$ ls
ackley_5.bch ex14_2_9.bch ex4_1_8.bch ex7_2_5.bch ex8_1_8bis.bch ex9_2_2.bch house.bch
chance.bch ex2_1_10.bch ex4_1_9.bch ex7_2_6.bch ex8_4inf-1.bch ex9_2_3.bch hs071.bch
ex14_1_1.bch ex2_1_1.bch ex5_2_2_case1.bch ex7_2_7.bch ex8_5_3-1.bch ex9_2_4.bch m4wd.bch
ex14_1_2.bch ex2_1_2.bch ex5_2_2_case2.bch ex7_3_1.bch ex8_5_3.bch ex9_2_5.bch pressure-vessel.bch
ex14_1_3.bch ex2_1_3.bch ex5_2_4.bch ex7_3_2.bch ex8_5_4.bch ex9_2_6.bch rastrigin.bch
ex14_1_4.bch ex2_1_4.bch ex5_4_2.bch ex7_3_3.bch ex8_5_5-1.bch ex9_2_7.bch rosenbrock10.bch
ex14_1_5.bch ex2_1_5.bch ex5_4_2bis.bch ex7_3_6.bch ex8_5_5.bch ex9_2_8.bch rosenbrock.bch
ex14_1_8.bch ex2_1_5bis.bch ex5_4_3.bch ex8_1_1.bch ex8_5_6-1.bch exbarron.bch schaffer.bch
ex14_1_9.bch ex2_1_6.bch ex6_1_2.bch ex8_1_2.bch ex9_1_10.bch exinfinity2.bch ship-1.bch
ex14_2_1.bch ex3_1_1.bch ex7_2_10.bch ex8_1_4.bch ex9_1_1.bch exinfinity3.bch test_infinity1.bch
ex14_2_2.bch ex3_1_2.bch ex7_2_11.bch ex8_1_5.bch ex9_1_2.bch exp40.bch test_infinity2.bch
ex14_2_4.bch ex3_1_3.bch ex7_2_1bis.bch ex8_1_6.bch ex9_1_4.bch f1.bch wall.bch
ex14_2_5.bch ex3_1_3bis.bch ex7_2_2.bch ex8_1_6bis.bch ex9_1_5.bch griewank.bch
ex14_2_6.bch ex3_1_4.bch ex7_2_3.bch ex8_1_7.bch ex9_1_8.bch himmel11.bch
ex14_2_8.bch ex4_1_5.bch ex7_2_3bis.bch ex8_1_8.bch ex9_2_1.bch hockschittkowski.mbx
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim/easy$
```

```
dualc1.bch ex8_4_5-1.bch notes.txt ship.bch test_infinity4.bch
ex14_1_6.bch ex8_4_5bis.bch polygon25.bch test_infinity3.bch
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim/others$
```

```
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim$ cd unsolved/
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim/unsolved$ ls
abel.bch chakra.bch ex5_3_3.bch ex8_1_3.bch ex8_4_2bis.bch least.bch qp1.bch water1.bch
abelbis-1.bch chenery-1.bch ex5_4_4.bch ex8_2_4.bch ex8_4_8.bch linear.bch qp2.bch water-abs.bch
abelbis.bch chenery.bch ex6_2_5.bch ex8_4_1.bch ex8_6_2.bch pollut.bch srcpm-1.bch water.bch
abelter-1.bch ex14_1_7.bch ex6_2_7.bch ex8_4_1bis.bch harker.bch prolog.bch srcpm.bch
chakra-1.bch ex5_2_5.bch ex7_2_9.bch ex8_4_2.bch immun.bch qp1-1.bch water1-abs.bch
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim/unsolved$
```

## 2. Análisis de los archivos de benchmark .bch

Se seleccionaron entre 3 y 5 archivos de cada carpeta antes mencionada, para cada archivo se realizó:

- Inspección de contenido con comando cat.
- Extracción manual de:
  - Número de variables.
  - Dominios de variables.
  - Forma de la función objetivo.
  - Tipo y cantidad de restricciones.
- Ejecución con ./ibexopt archivo.bch para así obtener:
  - cpu time used (tiempo de ejecución).
  - number of cell (nodos explorados).



```

valentina@valentina-Lenovo:~/ibex-lib/build/bin$ cat ../../benchmarks/optimization/design0/ex_3_4.bon
constants

variables
x1 in [-1.e8,1.e8];
x2 in [-1.e8,1.e8];
x3 in [-1.e8,1.e8];
x4 in [-1.e8,1.e8];
x5 in [-1.e8,1.e8];
x6 in [-1.e8,1.e8];
x7 in [-1.e8,1.e8];
x8 in [-1.e8,1.e8];
x9 in [-1.e8,1.e8];
x10 in [-1.e8,1.e8];
x11 in [0,18];
x12 in [-1.e8,1.e8];

minimize x12;
constraints
- x1 - x12 <= -10;
x1 - x12 <= 10;
x2 - 0.1*x12 <= 1 ;
- x2 - 0.1*x12 <= -1;
- x3 - 0.1*x12 <= -1;
x3 - 0.1*x12 <= 1;
- x4 - 0.01*x12 <= -0.2;
x4 - 0.01*x12 <= 0.2;
- x5 - 0.005*x12 <= -0.05;
x5 - 0.005*x12 <= 0.05;
x10*x11^4 - x8*x11^2 + x6 =0;
x9*x11^2 - x7 =0;

- 54.387*x3*x2 + x6 =0;
- 0.2*(1364.67*x3*x2 - 147.15*x4*x3*x2) + 5.544*x5 + x7 =0;

- 3*(-9.81*x3*x2^2 - 9.81*x3*x1*x2 - 4.312*x3^2*x2 + 264.896*x3*x2 + x4*x5 - 9.274*x5) + x8 =0;

- (7*x4*x3^2*x2 - 64.918*x3^2*x2 + 388.007*x3*x2 + 3*x5*x2 + 3*x5*x1) + x9 =0;

- x3^2*x2*(7*x1 + 4*x2) + x10 =0;

valentina@valentina-Lenovo:~/ibex-lib/build/bin$ 

```

```

x12 in [ 0, 1.e8];
x13 in [ 0, 1.e8];
x14 in [ 0, 1.e8];
x15 in [ 0, 1.e8];
x16 in [ 0, 1.e8];
x17 in [ 0, 1.e8];
x18 in [ 0, 1.e8];
x19 in [ 0, 1.e8];
x20 in [ 0, 1.e8];

minimize - 0.5*((x1 - 2)^2 + 2*(x2 - 2)^2 + 3*(x3 - 2)^2 + 4*(x4 - 2)^2 + 5*(x5 - 2)^2 + 6*(x6 - 2)^2 + 7*(x7 - 2)^2 + 8*(x8 - 2)^2 + 9*(x9 - 2)^2 + 10*(x10 - 2)^2 + 11*(x11 - 2)^2 + 12*(x12 - 2)^2 + 13*(x13 - 2)^2 + 14*(x14 - 2)^2 + 15*(x15 - 2)^2 + 16*(x16 - 2)^2 + 17*(x17 - 2)^2 + 18*(x18 - 2)^2 + 19*(x19 - 2)^2 + 20*(x20 - 2)^2);

constraints

- 3*x1 + 7*x2 - 5*x4 + x5 + x6 + 2*x8 - x9 - x10 - 9*x11 + 3*x12 + 5*x13 - 7*x14 - 7*x15 - 4*x16 - 6*x17 - 7*x18 - 4*x19 - 6*x20 <= -5;

7*x1 - 5*x3 + x4 + x5 + 2*x7 - x8 - 9*x9 + 3*x10 + 5*x11 + x14 + 7*x15 - 7*x16 - 7*x17 - 4*x18 - 6*x19 - 3*x20 <= 2;

- 5*x2 + x3 + x4 + 2*x6 - x7 - x8 - 9*x9 + 3*x10 + 5*x11 + x14 + 7*x15 - 7*x16 - 4*x17 - 6*x18 - 3*x19 + 7*x20 <= -1;

- 5*x1 + x2 + x3 + 2*x5 - x6 - x7 - 9*x8 + 3*x9 + 5*x10 + x13 + 7*x14 - 7*x15 - 4*x16 - 6*x17 - 3*x18 + 7*x19 <= -3;

x1 + x2 + 2*x4 - x5 - x6 - 9*x7 + 3*x8 + 5*x9 + x12 + 7*x13 - 7*x14 - 4*x15 - 6*x16 - 3*x17 + 7*x18 - 5*x20 <= 5;

x1 + 2*x3 - x4 - x5 - 9*x6 + 3*x7 - x8 + x11 + 7*x12 - 7*x13 - 4*x14 - 6*x15 - 3*x16 + 7*x17 - 5*x19 + x20 <= 4;

2*x2 - x3 - x4 - 9*x5 + 3*x6 + 5*x7 + x10 + 7*x11 - 7*x12 - 4*x13 - 6*x14 - 3*x15 + 7*x16 - 5*x18 + x19 + x20 <= -1;

2*x1 - x2 - x3 - 9*x4 + 3*x5 + 5*x6 + x9 + 7*x10 - 7*x11 - 4*x12 - 6*x13 + 7*x14 - 5*x17 + x18 + x19 <= 0;

- x1 - x2 - 9*x3 + 3*x4 + 5*x5 + x8 + 7*x9 - 7*x10 - 4*x11 - 6*x12 - 3*x13 + 7*x14 - 5*x16 + x17 + x18 + 2*x20 <= 9;

x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x20 <= 40;

valentina@Valentina-Lenovo:/ibex-lib/build/bin$ 

```

```

valentina@Valentina-Lenovo:/ibex-lib/build/bin$ cat ../../benchs/optim/constant.bch
constants

variables
x2 in [1.e-7, 0.5];
x3 in [1.e-7, 0.5];
x4 in [1.e-7, 0.5];
x5 in [1.e-7, 0.5];
x6 in [0, 1.e8];
x7 in [0, 1.e8];
x8 in [0, 1.e8];
x9 in [0, 1.e8];

minimize x2*(ln(x2) - ln(x2 + x4)) + x4*(ln(x4) - ln(x2 + x4)) + x3*(0.925356626778358*x2*x8 + 0.746014540096753*x4*x6 + 0.925356626778358*x5*x9);

constraints
x6*(x2 + 0.159040857374844*x4) - x2 =0;

x7*(x3 + 0.159040857374844*x5) - x3 =0;

x8*(0.307941026821595*x2 + x4) - x4 =0;

x9*(0.307941026821595*x3 + x5) - x5 =0;

x2 + x3 - 0.5 =0;

x4 + x5 - 0.5 =0;

end
valentina@Valentina-Lenovo:/ibex-lib/build/bin$ 
valentina@Valentina-Lenovo:/ibex-lib/build/bin$ 

```

```

absolute precision on f*: 0.0477528039408
cpu time used: 2072.95779801s
number of cells: 809930

.results written in ../../benchs/optim/hard/chem-1.csv
valentina@Valentina-Lenovo:/ibex-lib/build/bin$ cat ../../benchs/optim/hard/chem-1.bch

constants

variables
x1 in [0.001, 100];
x2 in [0.001, 100];
x3 in [0.001, 100];
x4 in [0.001, 100];
x5 in [0.001, 100];
x6 in [0.001, 100];
x7 in [0.001, 100];
x8 in [0.001, 100];
x9 in [0.001, 100];
x10 in [0.001, 100];
x11 in [0.01, 100];

minimize x1*(ln(x1/x11) - 6.05576803624071) + x2*(ln(x2/x11) - 17.1307680362407) + x3*(ln(x3/x11) - 34.0207680362407) + x4*(ln(x4/x11) - 5.88076803624071) + x5*(ln(x5/x11) - 24.6877680362407) + x6*(ln(x6/x11) - 14.9527680362407) + x7*(ln(x7/x11) - 24.0667680362407) + x8*(ln(x8/x11) - 10.6747680362407) + x9*(ln(x9/x11) - 26.6287680362407) + x10*(ln(x10/x11) - 22.1447680362407);

constraints
x1 + 2*x2 + 2*x3 + x6 + x10 - 2 =0;

x4 + 2*x5 + x6 + x7 - 1 =0;

x3 + x7 + x8 + 2*x9 + x10 - 1 =0;

- x1 - x2 - x3 - x4 - x5 - x6 - x7 - x8 - x9 - x10 + x11 =0;

end
valentina@Valentina-Lenovo:/ibex-lib/build/bin$ 

```

```

constants

variables

x2 in[ 1.e-6, 1];
x3 in[ 1.e-6, 1];
x4 in[ 1.e-6, 1];
x5 in [0,1.e8];
x6 in [0,1.e8];
x7 in [0,1.e8];

minimize x2*(0.28809 + ln(x2)) + x3*(ln(x3) - 0.29158) + x4*(0.59336 +
ln(x4)) + x2*(1.44805026165593*x6 + 0.989428667054834*x7) + x3*(1.12676386427658*x5 + 1.00363012835441*x7) + x4*(0.0347225450624344*x5 + 0.82681418300153*x6);

constraints

x5*(x2 + 0.145002897355373*x3 + 0.989528214945409*x4) - x2 =0;

x6*(0.293701311601799*x2 + x3 + 0.646291923054068*x4) - x3 =0;

x7*(0.619143628558899*x2 + 0.239837817616513*x3 + x4) - x4 =0;

x2 + x3 + x4 - 1 =0;

end
valentina@Valentina-Lenovo:~/ibex-lib/build/bin$
```

```

constants

variables
x2 in [0,1.e8];
x3 in [0,1.e8];
x4 in [0,20];
x5 in [0,20];
x6 in [0,20];
x7 in [0,20];
x8 in [0,20];
x9 in [0,20];
x10 in [0,20];
x11 in [0,20];

minimize (x2 - 5)^2 + (1 + 2*x3)^2;
constraints
- 3*x2 + x3 + x4 +3 =0;

x2 - 0.5*x3 + x5 - 4 =0;

x2 + x3 + x6 - 7 =0;

- x3 + x7 =0;

x4*x8 =0;

x5*x9 =0;

x6*x10 =0;

x7*x11 =0;

- 1.5*x2 + 2*x3 + x8 - 0.5*x9 + x10 - x11 - 2 =0;

end
valentina@Valentina-Lenovo:~/ibex-lib/build/bin$
```

```

constants

variables
x1 in [ 0, 100];
x2 in [ 0, 200];
x3 in[ 0, 500];
x4 in[ 0, 500];
x5 in[ 0, 500];
x6 in[ 0, 500];
x7 in[ 0, 500];
x8 in[ 0, 500];
x9 in[ 0, 500];

minimize - 9*x1 - 15*x2 + 6*x3 + 13*x4 + 10*x5 +
constraints
- x3 - x4 + x8 + x9 =0;

x1 - x5 - x8 =0;

x2 - x6 - x9 =0;

x7*x8 - 2.5*x1 + 2*x5 <= 0;
x7*x9 - 1.5*x2 + 2*x6 <= 0;

x7*x8 + x7*x9 - 3*x3 - x4 =0;

end
valentina@Valentina-Lenovo:~/ibex-lib/build/bin$
```

### 3. Estructura de un archivo .bch

Todos los archivos comparten una estructura común:

- **Declaración de variable:** Se especifican los nombres (x1, x2, x3,...) y sus dominios.
- **Función objetivo:** (minimize o maximize)
- **Restricciones:** (constraints)
- **Fin del archivo:** Cierra con la palabra clave **end**.

### 4. Ejecución con ibexopt

Para evaluar el rendimiento de cada problema y registrar métricas, se utilizó el comando:

```
./ibexopt ../../benchs/optim/{carpeta}/{archivo}.bch
```

Recordar: El objetivo de esto fue identificar patrones estructurales que pueden ser replicados mediante modelos generativos. Osea entrenar o guiar un modelo de IA para que genere problemas .bch nuevos pero similares en:

- cantidad y tipo de variables.
- complejidad de la función objetivo
- tipo y número de restricciones.
- dificultad medida por tiempo de ejecución y número de nodos

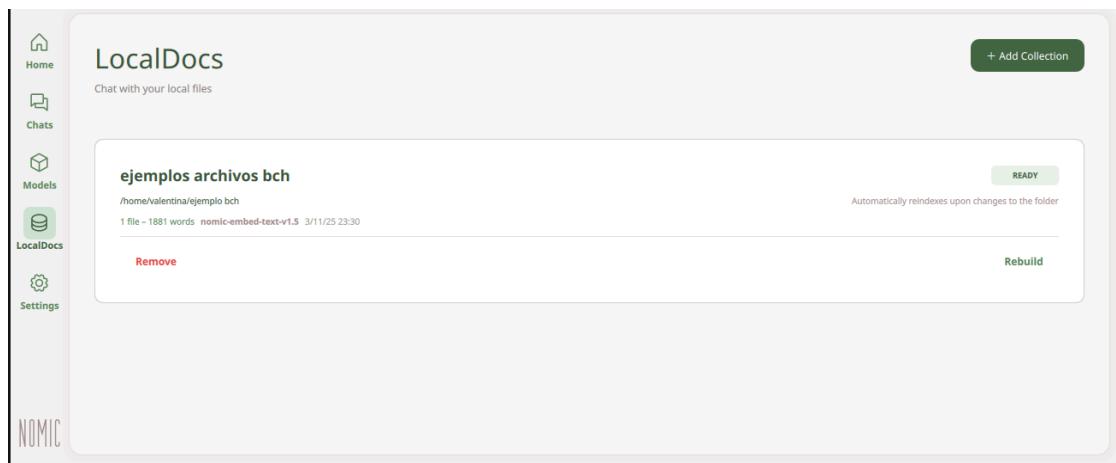
Estos nuevos ejercicios deberán ser exportados a archivos .bch, y luego validados usando ibexopt, para observar si el comportamiento (tiempo y nodos) se mantiene dentro de un rango similar a los problemas originales.

## Implementación de modelo generativo local.

### 1. Carga de ejemplos

Se creó un archivo de texto que contiene ejemplos reales de los archivos .bch del conjunto de pruebas de ibex.

Este archivo fue cargado en la sección LocalDocs de Gpt4all, lo que permite que el modelo analice y comprenda la estructura de los problemas de optimización (variables, dominios, función objetivo, restricciones y cierre con end).



## 2. Entrenamiento y comprensión del formato

El modelo Phi-3 Mini Instruct, ya instalado en gpt4all es un modelo generativo local entrenado previamente por Microsoft y adaptado para seguir instrucciones. En esta práctica, no se realiza un entrenamiento completo del modelo desde cero (fine-tuning), sino un entrenamiento contextual o aprendizaje basado en ejemplos mediante la carga de documentos locales en la sección LocalDocs.

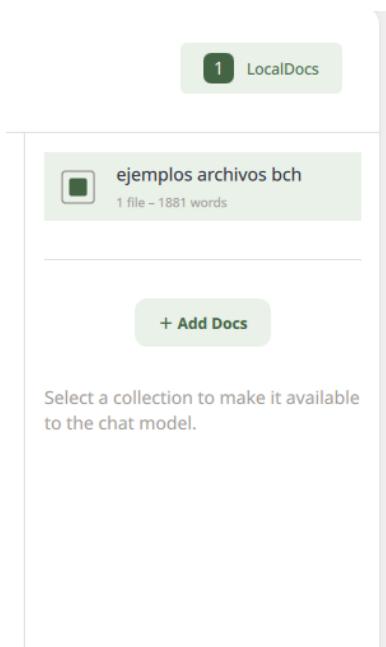
El objetivo de este proceso es que el modelo comprenda la estructura de los problemas de optimización (.bch) al analizarlos directamente desde el archivo .txt.

Entonces a partir de eso, el modelo aprende patrones sobre:

- formato del archivo (variables, constraints, minimize, end)
- cantidad y tipo de variables utilizadas.
- forma de las funciones objetivo.
- restricciones comunes.

Una vez comprendida la estructura, el modelo puede generar nuevos ejercicios con características similares, que posteriormente se exportan

y ejecutarán en ibexopt para verificar su validez y comportamiento (tiempo de cpu y número de nodos generados).



### 3. Ejercicio generados

El modelo se conectó con el archivo que contenía ejemplos previamente analizados del repositorio de ibex.

Se le solicitó que generaría un nuevo archivo con las siguientes condiciones:

You

Usa la colección llamada "ejemplos archivos bch" y genera un nuevo archivo .bch similar. Debe tener:

- 5 variables con dominios entre [0,100]
- una función objetivo de tipo cuadrática para minimizar
- 3 restricciones no lineales
- formato idéntico al de los ejemplos cargados (variables, función objetivo, constraints y end) El resultado debe ser texto plano con el formato exacto que pueda ejecutarse con ibexopt.

```
# ejemplo de archivo bch_5_variables.bch variables x1 in [ 0, 100]; x2 in
[0, 100]; x3 in [0, 100]; x4 in [0, 100]; x5 in [0, 100]; minimize - (x1^2 +
2*x2^2 + 3*x3^2 + 4*x4^2 + 5*x5^2); end #ejemplo de archivo
bch_5_variables.bch

# ejemplos archivos bch
Path: /home/valentina/ejemplo bch/ejemplos de archivo bch.txt
Excerpt: - 0.5*((x1 - 2)^2 + 2*(x2 - 2)^2 + 3*(x3 - 2)^2 + 4*(x4 - 2)^2 +
5*(x5 - 2)^2) <= 9;
2*x1 - x2 - x3 - 9*x4 + 3*x5 + 7*x6 + x8 + 2*x9 + 5*x10 - 7*x11 - 4*x12 -
6*x13 - 3*x14 + 7*x15 - 5*x16 + x17 + x18 + 2*x20 <= 9;
- x1 - x2 - 9*x3 + 3*x4 + 5*x5 + x8 + 7*x9 - 7*x10 - 4*x11 - 6*x12 - 3*x13 +
7*x14 - 5*x16 + x17 + x18 + 2*x20 <= 9;
x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12 + x13 + x14 +
x15 + x16 + x17 + x18 + x19 + x20 <= 40;
end #ejemplo de archivo bch_5_variables.bch
```

Como se ve en la imagen el modelo generó un texto con estructura similar a la esperada, pero con errores de formato y sintaxis:

- Se omitieron las palabras claves obligatorias (variables, constraints), y se incluyeron elementos adicionales (como encabezados, comentarios o nombres de ruta) que impidieron la correcta interpretación del archivo por ibexopt.

Para probar el resultado, se creó manualmente un archivo .bch dentro de la carpeta `/ibex/benchs/optim/generados/`, copiando el contenido generado por IA y corrigiendo los encabezados necesarios.

```
valentina@Valentina-Lenovo:~/ibex-lib/benchs$ cd optim/
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim$ mkdir -p generados
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim$ nano generados/ejemplo1gpt4all
valentina@Valentina-Lenovo:~/ibex-lib/benchs/optim$
```

```
GNU nano 7.2                                     generados/ejemplo1gpt4all
variables
x1 in [ 0, 100];
x2 in [0, 100];
x3 in [0, 100];
x4 in [0, 100];
x5 in [0, 100];

minimize - (x1^2 + 2*x2^2 + 3*x3^2 + 4*x4^2 + 5*x5^2);

constraints
- 0.5*((x1 - 2)^2 + 2*(x2 - 2)^2 + 3*(x3 - 2)^2 + 4*(x4 - 2)^2 + 5*(x5 - 2)^2) <= 9;
2*x1 - x2 - x3 - 9*x4 + 3*x5 + 7*x6 + x8 + 2*x9 + 5*x10 - 7*x11 - 4*x12 - 6*x13 - 3*x14 + 7*x15 - 5*x16 + x17 + x18 + 2*x20 <= 9;
- x1 - x2 - 9*x3 + 3*x4 + 5*x5 + x8 + 7*x9 - 7*x10 - 4*x11 - 6*x12 - 3*x13 + 7*x14 - 5*x16 + x17 + x18 + 2*x20 <= 9;
x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x20 <= 40;
end
```

Al ejecutar al archivo con IbexOpt, se obtuvo el siguiente mensaje de error:

```
valentina@Valentina-Lenovo:~/ibex-lib/build/bin$ ./ibexopt ../../benchs/optim/generados/ejemplo1gpt4all
Syntax error near "x6" line 12: syntax error
```

Esto se debió a que el modelo utilizó variables ( $x_6$ ,  $x_7$ , etc). que no habían sido previamente declaradas, lo cual confirma que la IA comprende parcialmente la estructura del archivo, pero no su coherencia interna.

## Segundo intento - refinamiento del modelo

A partir del resultado anterior, se procedió a realizar un nuevo intento, esta vez entregando al modelo un prompt más detallado, con reglas explícitas sobre la sintaxis y formato del archivo.

You

Quiero que generes un nuevo archivo .bch válido para IbexOpt, basándote en los ejemplos de mi colección llamada ejemplos archivos bch.

Requisitos estrictos:

Escribe solo el contenido del archivo, sin explicaciones, sin comentarios ni texto adicional.

Sigue exactamente esta estructura: variables x1 in [a,b]; ... minimize ( expresión matemática ); constraints restricciones; end

Termina cada línea con ;.

No pongas texto antes o después del bloque.

Condiciones del ejercicio:

5 variables (x1..x5) con dominios entre [0,100].

Función objetivo cuadrática para minimizar.

3 restricciones: 2 no lineales y 1 de igualdad.

todas las variables utilizadas en las restricciones deben estar declaradas en la sección de variables

Si no puedes cumplir exactamente el formato, repite el intento hasta que todas las variables estén declaradas y la sintaxis sea válida.

🔗

GPT4All Phi-3 Mini Instruct searching localdocs: ejemplos archivos bch ...

En este segundo intento, el modelo logró generar un archivo .bch estructuralmente correcto, que cumple con los requerimientos básicos de ibex.

El archivo fue ajustado mínimamente para respetar la sintaxis de ibex (se borro un signo “=” que el modelo al generarla agrego uno demás).

El resultado final fue exitosamente interpretado por ibexopt, lo cual demuestra que el modelo comprende la estructura fundamental de los archivos de optimización, alcanzando el objetivo de esta etapa: generar problemas válidos a partir de ejemplos.

 GPT4All Phi-3 Mini Instruct

```
variables x1 in [0, 100]; x2 in [0, 100]; x3 in [0, 100]; x4 in [0, 100]; x5 in [0, 100]; minimize - (x1^2 + x2^2 + x3^2 + x4^2 + x5^2); constraints x1*(4*x1 - 2*x2 + 2*x3) + x2*(2*x2 - 2*x1 - x3) + x3*(2*x1 - x2 + 2*x3) >= -24; x1 + x2 <= 4; 3*x2 + x3 == 6; end
```

1 Source >

⟳ ⟲

```
valentina@Valentina-Lenovo:~/ibex-lib/build/bin$ ./ibexopt ../../benchs/optim/generados/ejemplo2gptall
*****
file loaded:      ../../benchs/optim/generados/ejemplo2gptall
output COV file: ../../cov
*****
running.....
optimization successful!

f* in [-1.00111523348e-07, -1.11523348749e-10]
(best bound)

x* = (0 ; 0 ; 1.15829621504e-11 ; 4.38901526556e-12 ; 5.25408923964e-126)
(best feasible point)

relative precision on f*: 0.998886008873
absolute precision on f*: 1e-07 [passed]
cpu time used: 7e-06s
number of cells: 0

results written in ../../cov
```

En esta fase se elaboró un *prompt definitivo* para el modelo **Phi-3 Mini Instruct**, el cual integra todas las reglas sintácticas y estructurales aprendidas durante los intentos previos.

El objetivo fue que el modelo produjera ejercicios completamente válidos y ejecutables en **IbexOpt** sin intervención manual.

Este ajuste permitió obtener archivos con estructura correcta, funciones cuadráticas bien definidas y restricciones coherentes, logrando finalmente la automatización completa del proceso de generación.

### You

Quiero que generes un archivo .bch completamente válido para IbexOpt, con el mismo formato que los ejemplos de mi colección local ejemplos archivos bch.

□ Requisitos de formato

Escribe solo el contenido del archivo, sin texto adicional ni explicaciones.

Usa esta estructura exacta: variables x1 in [a,b]; ... minimize (expresión); constraints restricción1; restricción2; ... end

Cada línea termina con ;.

Las variables deben estar numeradas secuencialmente (x1, x2, ...).

Todas las variables usadas en restricciones deben estar declaradas arriba.

No incluir texto fuera del bloque (ni "Path", ni "Excerpt", ni "# ...").

Condiciones del ejercicio

5 variables con dominios entre [0,100].

Función objetivo cuadrática para minimizar.

3 restricciones: dos no lineales y una de igualdad.

Las constantes y coeficientes deben ser enteros o decimales simples (sin fracciones ni texto).

Validación final

Verificar el archivo

Antes de mostrar el resultado, verifica internamente que el archivo cumple:  
Todas las variables usadas están declaradas.  
La sintaxis es compatible con Ibex (sin errores de formato).  
Hay un solo bloque variables, un solo bloque constraints y un end.  
Si algo no cumple, vuelve a generar otro hasta que sea completamente correcto.  
Devuélvelo entre triple comillas

bch para copiarlo fácilmente.



GPT4All Phi-3 Mini Instruct searching localdocs: ejemplos archivos bch ...

...

El resultado en su mayoría fue exitoso pero nuevamente el modelo entrega un resultado donde el ejercicio contiene dos signos == haciendo que ibexpot lo detecte como error.

□ Phi-3 Mini Instruct □

🕒 GPT4All Phi-3 Mini Instruct

```
variables
x1 in [0, 100];
x2 in [0, 100];
x3 in [0, 100];
x4 in [0, 100];
minimize (5*x1 + x2 - 6.789*x3^2 + 2.345*x4);
constraints
- 2*x1 + 3*x2 <= 4;
- ln(x3/0.1) - 5*x4 >= 3;
x1 + x2 == 75;
end
```

🕒 1 Source >

⟳ □

...

✉️ Send a message... ➔

Selected to t

```
valentina@Valentina-Lenovo:~/ibex-lib/build/bin$ nano ejemplo3gptall
valentina@Valentina-Lenovo:~/ibex-lib/build/bin$ ./ibexopt ../../benchs/optim/generados/ejemplo3gptall
Error: cannot read file '../../benchs/optim/generados/ejemplo3gptall'
```

En la consola se procede arreglar dicho problema.

```
GNU nano 7.2
variables
x1 in [0, 100];
x2 in [0, 100];
x3 in [0, 100];
x4 in [0, 100];
minimize (5*x1 + x2 - 6.789*x3^2 + 2.345*x4);
constraints
- 2*x1 + 3*x2 <= 4;
- ln(x3/0.1) - 5*x4 >= 3;
x1 + x2 = 75;
end
```

Se intentó nuevamente

```
variables
x1 in [0,5]; x2 in [0,5]; x3 in [0,5]; x4 in [0,5]; x5 in [0,5];
minimize (x1^2 + 2*x2^2 + 3*x3^2 + 4*x4^2 + 5*x5^2 - 6*x1 - 7*x2);
constraints
- x1 * x2 <= 0.8;
(sin(pi/4)*sqrt(x3)) + (cos(pi/6)*sqrt(x4)) >= sqrt(x5) - 1;
(-x1)^2 + (-x2)^2 + x3^2 + x4^2 + x5^2 <= 0.8 * (x1*x2);
end
```

Teniendo éxito, ya que no hubo que modificar nada.

```

valentina@Valentina-Lenovo:~/ibex-lib/build/bin$ cd ~/ibex-lib/build/bin
./ibexopt -t 20 -r 1e-2 -a 1e-6 --simpl=1 ../../benchs/optim/generados/ejemplo4gptall \
| tee ../../benchs/optim/generados/ejemplo4gptall_log.txt

***** setup *****
file loaded:      ../../benchs/optim/generados/ejemplo4gptall
rel-eps-f:        0.0100001    (relative precision on objective)
abs-eps-f:        1.00001e-06   (absolute precision on objective)
symbolic simpl level: 1
output COV file:  ../../cov
timeout:          20s
*****
running.....
time limit 20s. reached

f* in [-4.08649224022e-161,inf]
(best bound)

x* = --
(no feasible point found)

relative precision on f*:      inf
absolute precision on f*:      inf
cpu time used:                19.9995000001s
number of cells:               66364

results written in ../../cov
(old file saved in ../../cov~)

```

Para evitar tiempos excesivos, se fijó un timeout de 20 s (-t 20), precisión relativa (-r 1e-2), precisión absoluta -a 1e-6 y simplificación -simpl=1. Se redirigió el output con tee para registrar métricas.

### 3. Cambiar el objetivo de los prompts

Hasta ahora las indicaciones han sido muy precisas, esto fue necesario para que el modelo aprendiera una estructura base. Ahora que ya aprendió dicho patrón, el siguiente paso es soltar un poco el control.



Genera un nuevo problema de optimización no lineal válido para IbexOpt, manteniendo el formato correcto (variables, minimize, constraints, end), pero varía libremente el número de variables (entre 3 y 7) y el tipo de restricciones. La función objetivo debe seguir siendo para minimizar, pero puede incluir combinaciones cuadráticas, raíces, senos o cosenos



GPT4ALL Phi-3 Mini Instruct searching localdocs: ejemplos archivos bch ...

## 5. Entrenamiento del modelo local con ejemplos válidos

Para mejorar la generación de problemas .bch por parte del modelo Phi-3 Mini Instruct, se realizó una carga de ejemplos válidos provenientes de la librería IBEX, ubicada en la ruta:

```
~/ibex-lib/benchs/optim/easy/
```

Estos archivos contienen problemas de optimización correctamente estructurados y funcionales, que sirven como base para que la IA reconozca patrones en la sintaxis de IBEX (declaración de variables, definición de la función objetivo y restricciones).

Sin embargo, el entorno GPT4All no reconoce directamente archivos con extensión .bch, por lo que fue necesario convertirlos temporalmente a texto plano (.txt).

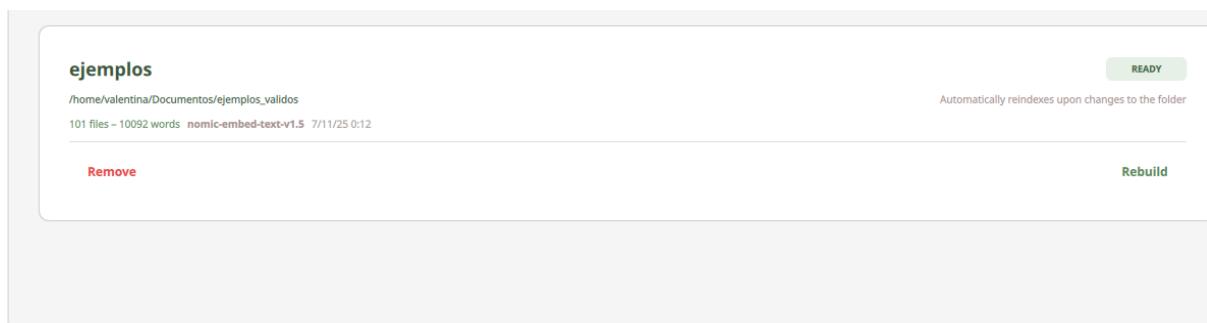
Para esto, se utilizó el siguiente comando en la terminal:

```
for f in *.bch; do cp "$f" "${f%.bch}.txt"; done
```

Con ello se generó una carpeta con 101 archivos válidos en la ruta:

```
/home/valentina/Documentos/ejemplos_validos
```

Posteriormente, esta carpeta fue indexada en GPT4All como colección local denominada “ejemplos”, tal como se observa en la imagen donde se muestra el proceso de *embedding* (lectura e indexación de texto por el modelo).



Este proceso permitió que el modelo local asimilara la estructura formal de los problemas .bch, lo que facilita su aprendizaje contextual y mejora la calidad de los ejercicios generados posteriormente.



You

Genera un archivo .bch válido para ibexopt. Reglas estrictas:

Usa exactamente estas variables y ninguna otra: {x1, x2, x3, x4, x5}.

Declara dominios: x1..x5 in [0, 100];

Una sola función objetivo: minimize <expresión>;

Luego la sección constraints con tres restricciones:

1 de igualdad (usa = de IBEX, NO ==)

1 de " $\leq$ "

1 de " $\geq$ "

Prohibido usar variables no declaradas.

Si usas ln(), sqrt() o división, asegura el dominio en las restricciones (p.ej.  $x1>0$ ,  $x3!=0$ , etc.).

Un enunciado por línea, termina con end y no pongas texto extra.

Devuélveme sólo el contenido del archivo (sin comentarios ni "Collection/Path/Excerpt").

Ejemplo de estructura a seguir:

variables x1 in [0,100]; x2 in [0,100]; x3 in [0,100]; x4 in [0,100]; x5 in [0,100]; minimize ... ; constraints

\* ... = ...;

\* ...  $\leq$  ...;

\* ...  $\geq$  ...; end

Entrega ahora un archivo con esa estructura.



GPT4ALL Phi-3 Mini Instruct

variables x1 in [0, 100]; x2 in [0, 100]; x3 in [0, 100]; x4 in [0, 100]; x5 in [0, 100]; minimize -0.8x1^2 + x2; constraints -0.8x1  $\leq$  79.2; x3  $\geq$  0; ln(x4)  $>$  0;

3 Sources >



```
valentina@Valentina-Lenovo:~/ibex-lib/build/bin$ ./ibexopt ../../benchs/optim/generados/ejemplo5.2
Syntax error: not a system
valentina@Valentina-Lenovo:~/ibex-lib/build/bin$
```

Genera SOLO el contenido de un archivo .bch válido para ibexopt. NO incluyas "Collection:", "Path:", "Excerpt:" ni texto extra. Respeta EXACTAMENTE este formato y sintaxis (con \* y ;):

variables x1 in [a1, b1]; x2 in [a2, b2]; ... xN in [aN, bN];

minimize (expresión matemática usando x1..xN);

constraints (inecuación o ecuación 1 con  $\leq$ ,  $\geq$  o  $=$  usando \* y ; ) (inecuación o ecuación 2 con  $\leq$ ,  $\geq$  o  $=$  y ; ) ... (opcionalmente más)

end

Reglas estrictas:

- SIEMPRE usa \* para multiplicar (ej.:  $5x1$ ,  $0.5x2^2$ ).
- Cierra cada línea con ;
- Solo  $\leq$ ,  $\geq$  o  $=$  (no uses  $>$  ni  $<$  estrictos).
- Toda variable usada aparece declarada en "variables".
- Termina con "end" al final en su propia línea.

Ejemplo 1 (válido): variables x1 in [0, 100]; x2 in [0, 100]; x3 in [0, 100];

minimize ( $x1^2 + 2x2^2 - 3x3 + 5$ );

constraints  $x1 + x2 - 2x3 \geq 0$ ;  $\sin(x1) + 0.1x2 \leq 2$ ;  $x3 = 10$ ;

end

Ejemplo 2 (válido): variables x1 in [0, 5]; x2 in [0, 5]; x3 in [0, 5]; x4 in [0, 5]; x5 in [0, 5];

minimize ( $x1^2 + 2x2^2 + 3x3^2 + 4x4^2 + 5x5^2 - 6x1 - 7x2$ );

constraints

•  $x1*x2 \leq 0.8$ ;  $(\sin(\pi/4)\sqrt{x3}) + (\cos(\pi/6)\sqrt{x4}) \geq \sqrt{x5} - 1$ ;  $(-x1)^2 + (-x2)^2 + x3^2 + x4^2 + x5^2 \leq 0.8(x1*x2)$ ;

\*\*\*

end

Ahora genera un nuevo .bch con:

- 5 variables x1..x5 con dominios en [0, 100]
- función objetivo cuadrática (signo y coeficientes libres)
- 3 restricciones (2 no lineales y 1 de igualdad)
- Todo conforme a las reglas estrictas anteriores. Devuelve SOLO el código .bch.

