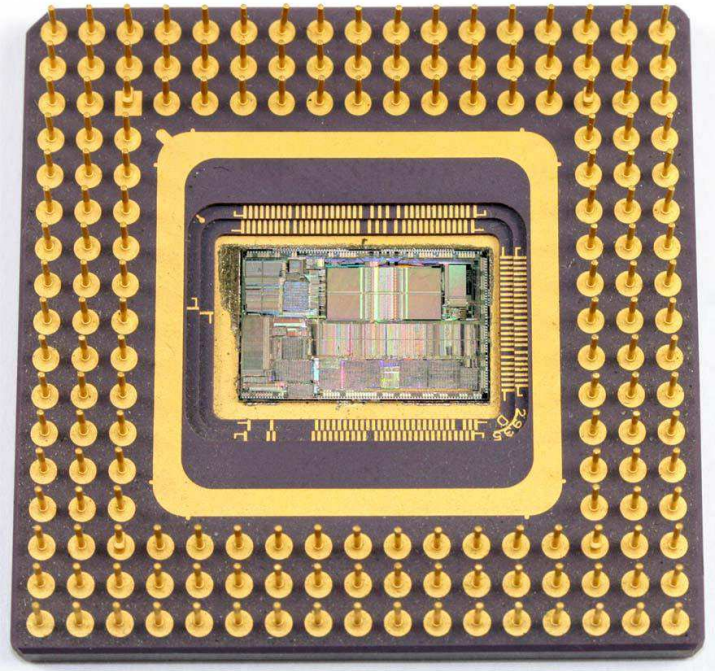
# El test



**S**e quedó mirando la esfera plateada de su Rolex Milgauss, mientras la segundera avanzaba casi de manera continuada, a intervalos de 181,81 milisegundos periódicos. Sin lugar a dudas, era un reloj que, de manera razonable, no podía permitirse. Sin embargo, cuando la marca ginebresa decidió detener su producción, se le presentó la oportunidad de comprarlo a buen precio.

De alguna forma, este embelesamiento le centraba en sus pensamientos, la estilizada aguja en suave movimiento circular (antes que decidieran ponerle una en forma de rayo), era como si separase sus ideas.

**…**

**Q**uedaba muy lejano aquel año 1952 en el que el EDSAC (Electronic Delay Storage Automatic Calculator) consiguió partidas perfectas al Tres en Raya, venciendo sin miramientos a los contrincantes humanos. Contaba con 3.000 válvulas que ocupaban una habitación de 5mx4m al completo, y absorbían energía al ritmo de 12 kW/h. Sin lugar a dudas, aquello estaba más que superado desde hacía décadas. Un simple Altair 8800 de 1972, que gozaba del honor de ser el primer ordenador doméstico, ya excedía la capacidad de proceso del EDSAC en un factor de más de 400.

Parecía que el logro de EDSAC, iba a cambiar el mundo, las personas dejaríamos de jugar al Tres en Raya, al considerarlo un juego trivial. Por supuesto, esto no ocurrió jamás, del mismo modo que los niños siguen imaginando una Luna formada por queso, y los enamorados ven en ella su halo romántico y sobrenatural, que nada tiene que ver con el satélito que pisó Neil Armstrong en 1969.

Sin embargo, el mayor reto, había quedado formulado desde algunos años atrás, en concreto por Alan Turing en el año 1950. Se trataba del que se conocería como Test de Turing, una prueba en la que la máquina debería demostrar tanta inteligencia como un humano. Es decir, ambos deberían ser indistinguibles. Para ello se basaba en un conocido juego de imitación, en el que la idea original fuera tener a tres personas: un interrogador, un hombre y una mujer. El interrogador estaría separado de los otros dos, y sólo podría comunicarse con ellos mediante un lenguaje que todas las partes entendieran. El objetivo del interrogador sería descubrir quién era la mujer y quien era el hombre, mientras que el de los otros dos, sería convencer al interrogador de que son la mujer, o sea, de engañarle.

El añadido de Turing, fue alterar el objetivo del juego. En vez de hombre o mujer, sería el de humano o máquina. De esta manera, si el interrogador humano era capaz de averiguar al menos el 70% de las veces, y en el límite de 5 minutos o menos, quién era el hombre/mujer, y quien la máquina, la prueba fracasaba, y se daba por concluido que la máquina no pensaba. En caso contrario, se determinaría que la máquina era inteligente, o al menos, no diferenciable en la prueba, de una inteligencia humana.

**…**

**C**abría pensar que, con tantos avances en lo relativo velocidad de proceso, lo habríamos logrado hace muchos años. Sorprendentemente, no fue así, estábamos en 1990, y nadie lo había conseguido… Aún.

Desde mi punto de vista, llamadlo deformación profesional si queréis, el problema estribaba en las implementaciones. Es decir, en cómo había que escribir un programa capaz de superar el Test de Turing.

Todos conocíamos a Eliza, el programa para el IBM 7094 que, en 1964, Joseph Weizenbaum escribió en el MIT. Era tremendamente rudimentario, pero sentó las bases que el resto habríamos de seguir años después.

El problema, es que estaban equivocados… Cuando optaban por LISP o PROLOG, los lenguajes de programación por antonomasia en el ámbito académico de la Inteligencia Artificial, estaban desaprovechando una gran parte de la capacidad del ordenador. O sea, conseguían hacer mucho menos, de lo que en realidad podrían hacer. Lo mismo ocurriría después con BASIC y FORTRAN, lenguajes orientados al mundo científico, pero que en ningún momento estaban planteados para exprimir todo el potencial disponible.

Por eso mi caso era diferente. Había decidido utilizar lenguaje C, el legendario lenguaje de programación creado por Dennis Ritchie en 1972, y que con el objetivo de crear el sistema operativo UNIX, estaba pensado precisamente para sacar todo el jugo a los recursos computacionales disponibles.

Claro que, hacer ese tipo de proyecto en C, era mucho más complicado, arduo y laborioso que los que optaban por LISP. No obstante, yo estaba seguro de ello, y tras algunos meses de intenso trabajo con Watcom C 7.0 en mi reluciente ordenador clónico 486 con 4 MB de RAM, lo había conseguido.

Mi creación, a la que llamé Jack, era capaz de realizar análisis sintácticos de lo que introducía el usuario por teclado. Ello permitía, en definitiva, que la máquina comprendiera su contenido. Todo ello, a la velocidad de los 32 bits de proceso de la que era capaz.

Una vez analizado, y separado sintácticamente el contenido, era sencillo cambiarle la forma personal o el género, lo que le permitía, utilizar las entradas del usuario, como respuestas del programa. Imaginaros esto:

- **Usuario**: Lisa es mi mejor amiga.
- **Jack**: Los amigos son algo muy importante.
  …
- **Usuario**: ¿Quién es Lisa?
- **Jack**: Lisa es tu mejor amiga.

¿Os habéis percatado? Jack ha sido capaz de “aprender” quien es Lisa, y dar al final de la conversación la respuesta de que es el mejor amigo del usuario.

Un truco que utilicé, es el de tener una batería de frases comodín, es decir, expresiones tan comunes que no aportan nada a la conversación, salvo darle continuidad, y en cierta forma, asemejarse a un diálogo. Es el ejemplo de “Los amigos son algo muy importante”. Tenía cargadas decenas de frases comodín, y Jack simplemente las extraía de manera aleatoria para engañar al ingenuo humano.

Ahora mismo, lector, estarás infravalorando mi esfuerzo. Lo sé. Quizás pienses que este ejercicio es algo que un niño de 3 años puede hacer con relativa facilidad. Y, debería darte la

razón. Es algo que es muy sumamente fácil para un niño. El motivo, es que un niño es humano, y en esos 3 años, ha sido capaz de aprender un vocabulario, y asimilar unos conceptos, que de manera innata le permite conocer los elementos que forman parte de la oración.

"Lisa es mi mejor amiga", implica que al menos conozcamos que Lisa es un nombre (propio en este caso) que hace de sujeto, que "es", actúa de verbo, y que "mi mejor amiga" es un atributo como parte del predicado.

La forma más sencilla de hacer esto, era manejar una lista de verbos, tanto en su forma regular, con las directrices necesarias para conjugarlos, como en su forma irregular. Al menos en los más habituales. Con ello Jack, era al menos capaz de separar frases, y por tanto asociar términos como habéis podido ver.

Me sentía orgullo de la velocidad en que todo Jack era capaz de procesar esas operaciones, y otras mucho más complejas. Iba a ser el primer programador que superaría el Test de Turing. No me haría rico, pero me daría algo de fama, y me haría sentir muy bien, habiendo conseguido algo, que nadie había conseguido hasta entonces. Algo en mi esperaba que ese logro hiciera que el mundo fuera un sitio mejor para muchos. Quizás acercando las máquinas a las personas, haciéndolas más listas, y más sensibles a nuestras necesidades.

**…**

**N**o era suficiente… Desgraciadamente, no lo era. Había trabajado tan duro en Jack, que perdí de vista el objetivo. Claramente era el más veloz analizando frases, consultando listas, y procesando archivos de datos. Pero su objetivo era comportarse como un humano. Algo que no lograba:

- **Usuario**: ¿Quién es George H. W. Bush?
- **Jack**: Cuéntame más de George H. W. Bush.

Ahí estaba. Otra de las frases comodín de cuando Jack no tenía ni idea de lo que decir. El único mérito, era su eficiente procesado de cadenas de caracteres, y haber concluido que "George H. W. Bush" era el predicado verbal. Claro, por más que buscase en su base de datos de nombres, no aparecería nada relacionado con George Bush, así que se quedaba sin respuesta, y optaba por el comportamiento programado de la frase que vale para todo.

Y así seguía, mirando la esfera de mi Rolex, viendo el tiempo pasar, mientras que mi mente intentaba desentrañar el rompecabezas. ¿Cómo seguir? ¿Cómo hacer que Jack supiera quien es George H. W. Bush?

**…**

**A**hora entiendo lo que sintió Arquímedes con su ¡Eureka! Ya lo tenía. Era solamente cuestión de atiborrar de contenido al sistema. Algo que le dijera quien es "George Bush".

Desde los tiempos de la Revolución Francesa con Denis Diderot, todos conocemos la Enciclopedia, ese compendio que recopila el conjunto del saber. Vale, pues quizás no sepas

que antes de Diderot estuvo Ephraim Chambers, pero eso da igual. ¡Una enciclopedia, era todo lo que Jack necesitaba!

Me hice con la Compton's Multimedia Encyclopedia, y un lector de CD-ROM, algo que os puedo asegurar no fue ni barato, ni sencillo de conseguir, y enlacé a Jack con esas definiciones. Allí estaban todos los verbos y los nombres, que él necesitaba. O si no estaban todos, sí que figuraban muchos más de los que la mayoría de personas conocemos.

- **Usuario**: ¿Quién es George H. W. Bush?
- **Jack**: Es el 41.º Presidente de los Estados Unidos. ¿Por qué lo preguntas?

¡Genial! Sencillamente alucinante. Nutriéndose de la enciclopedia de Compton, Jack sabía lo que era necesario saber. En este caso, que Bush era nuestro presidente. De nuevo la cuestión de "¿Por qué lo preguntas?", era una frase comodín más. Una idea brillante, aunque esté mal que yo lo diga, que de nuevo pretendía dar continuidad a la simulación, y lo que era más importante, devolver la pelota al usuario.

Ya sabéis, cuanto más tiempo hable el entrevistador, menos lo hará Jack, menos posibilidades de ser descubierto, y más posibilidades de exceder el límite de 5 minutos que fijó Turing.

Sonaba muy bien, pero había otro problema. El precio a pagar por los 600 MB de información que tenía la Compton, era lo lento de su acceso. Con un tiempo de respuesta medio de 600 milisegundos, y una velocidad de acceso máxima de 150 KB/segundo, las operaciones de Jack, tardaban cerca de 5-10 segundos en completarse. Me refiero solamente a las operaciones con el CD-ROM, buscar las definiciones de las piezas clave de la frase, procesar y tratar las respuestas… Imaginaros una conversación así, en que vuestro interlocutor tarde hasta 10 segundos en responder. ¡Era más tiempo del que tardaría la respuesta en ir y volver de la Luna!

Quizás si la frase fuera sencilla, el algoritmo tuviera suerte, y el caché de DR-DOS 5 hubiera guardado esa información en la rápida memoria RAM. Entonces Jack respondería en 2 segundos. Considerando que una persona lo hace en entre 0,25 segundos, y 1 segundo, dependiendo de lo sencilla que sea la respuesta, y de lo despierta que esté la persona (a eso se le llama tiempo de reacción complejo), el comportamiento de Jack podría tener un pase. Pero claro, con medias de 5 a 10 segundos, el diálogo no sería fluido, y empezaría a levantar sospechas en el interrogador de inmediato.

**…**

**P**or suerte, acelerar cosas, no era sólo lo que más me gustaba (¿recuerdas lo de lenguaje C?), sino que además era increíblemente bueno en ello. Se trataba de aprovechar mejor lo que había disponible, para hacer más. Y siendo sinceros, era necesario no sólo hacer más, sino mucho más.

La primera idea, fue copiar el contenido del CD-ROM al completo en el disco duro. Descartada. Imposible meter casi 650 MB de información del CD en un disco fijo de 250 MB. Conseguir un disco duro más grande, tampoco era viable, quizás uno de 320 MB fuera asequible, pero 650 MB. Era imposible.

Me deshice de todos los recursos multimedia que tenía el CD-ROM, no necesitaba ni imágenes ni audio, con la información textual era suficiente. Se había quedado ahora en unos 400 MB. Una buena reducción, pero era insuficiente. Aún no entraría en mi disco duro.

No quedaba otra que intentar comprimir el contenido, y así ahorrar espacio hasta que cupiera. Tampoco funcionaba, ese contenido estaba ya comprimido, y ya sabéis que no se puede comprimir algo que ya está comprimido. Bueno, no es así exactamente como lo dijo Shannon, pero para que me entendáis.

Haciendo ingeniería inversa, conseguí discernir el algoritmo de compresión que usaba Compton, como cabría esperar, era una variante de LZ. Me quedé pasmado esos 400 MB de datos comprimidos, se extendían a cerca de 1000 MB una vez descomprimidos. Un Gigabyte (GB). ¿Habíais oído alguna vez esta medida?

Decidí confiar en mí, había leído algo de PPM (Prediction by Partial Matching), un algoritmo de compresión de notable eficacia, especialmente en contenidos de texto, que es lo que necesitaba. Esperaba que con PPM consiguiera reducir esos 1000 MB a una cifra manejable por mi disco duro. Lo que iba a hacer sería escribir un programa que descomprimiera el LZ del CD_ROM original, y entonces lo comprimiera con PPM a mi disco duro. De nuevo, teníamos un reto. Si mi disco duro era de 250 MB, ¿Cómo iba a meter esos 1000 MB de datos descomprimidos, para luego volverlos a comprimir en el mismo lugar? Si las cuentas no fallaban, necesitaría tener del orden de 1400 MB disponibles, y sólo tenía 250 MB en el mejor de los casos.

A grandes males, grandes remedios, podía irlo haciendo por letras del alfabeto. Leería todas las palabras que comienzan por A, las descomprimiría, y luego las archivaría con mi sistema PPM. De esta forma, el espacio ocupado por todas las palabras se iría reduciendo paulatinamente.

Como de costumbre, no era el escenario ideal. Hubiera sido mucho más eficiente, comprimir todas las letras juntas. Así aprovechamos redundancias, y se obtienen mejores resultados. Piensa por ejemplo en la palabra Abeto. Tiene una letra A, que se comprimiría muy bien con el resto de As, en cambio luego le sigue una B, por lo que, si pudiera estar junto a las que empiezan por B, ahorraríamos todavía más. No podía ser, como se dice a menudo, cada uno debe jugar con las cartas que le han tocado, y no podía meter todas las letras juntas.

Lo bueno, es que ese sistema me serviría de índice. Es decir, que sabiendo donde se hubiera guardado la letra B, podría ir allí directamente a buscar la palabra “Bush”. No tendría que “preguntar” al sistema: “Oye, ¿Dónde dijiste que se guardó la B?”.

Por cierto, que PPM tenía un grave inconveniente, era asimétrico. Esto quiere decir que, aunque descomprimía con mucha velocidad, que es lo que yo necesitaba para que Jack pudiera acceder a la información a la velocidad del rayo, comprimía muy lento. No os quiero aburrir, pero mi súper 486 a 33 Mhz, el mejor PC que puedas encontrar a la venta, estuvo casi 3 días comprimiendo letra por letra la enciclopedia entera.

**…**

**C**uando terminó y vi la pantalla: ¡Madre mía! Los datos quedaron reducidos a sólo 160 MB. De los 400 MB originales, eso era menos de la mitad, sin eliminar nada. Justo la misma

información, en menos de la mitad de sitio. O, dicho de otro modo, si Compton quisiera, podría guardar más del doble de datos. Desde luego, tendría que escribir a Britannica (los que compraron Compton), cuando terminara con esto. ¿Quién sabe? Igual mi futuro cambiase. Lo importante de momento, es que dio resultado, y eran suficientes para que cupieran en mi disco duro. Incluso me sobraba sitio para el sistema operativo, mis herramientas de desarrollo y utilidades, proyectos como Jack, y otras tablas de datos.

El final fue lo más fácil, integrar la descompresión PPM en Jack, y usar los índices alfabéticos para acceder a los conceptos que hicieran falta. Usando el cronográfo del Casio DW-5600C que usaba antes, y que todavía guardaba en el cajón de mi despacho, con la pila perfectamente operativa, empecé a hacer unos juegos de prueba. Todo era perfecto.

- **Usuario**: ¿Dónde está Hollywood?
- **Jack**: Hollywood está en California. ¿Te gusta Hollywood?

Magnífico, usando la conexión con la Compton, había conseguido acceder a Hollywood, de ahí saber que pertenece a la ciudad de Los Ángeles (desde 1903), y que a su vez Los Ángeles pertenece al estado de California. Varias consultas a los datos comprimidos, y según el reloj digital, habían transcurrido solamente 0,49 segundos. Para que me entendáis, sin retraso perceptible en la respuesta. Similar al de una persona que no conociera en profundidad los límites políticos de Estados Unidos tendría.

Ahora sí, Jack podía enfrentarse ante el Test de Turing… Y superarlo.

**FIN**

**NOTAS**

La historia se me ocurrió a principios de 2013, y la empecé a desarrollar. Lo hice ambientado en un futuro cercano, y en realidad trataba de una máquina ocupando el papel de interrogador. Es decir, tratando de determinar si los sujetos eran humanos o máquinas. Vendría a ser el Test de Turing Redux. Imagino que sabes que, en 2014, Eugene superó el Test de Turing. Antes de eso Deep Blue venció en ajedrez al mejor jugador del momento (1997), y Chinook consiguió vencer a las damas (1994).

La dejé a medias, la trama no daba el juego que buscaba. De repente a finales de 2016, se me ocurrió la idea de ambientarla en el pasado. En la época del apogeo de DOS, y el crecimiento de las capacidades de proceso. Me puse, y la escribí de un tirón en 4 horas. Verás que tiene bastantes reseñas científicas, y he intentado que todas sean verdad. Seguramente en Compton, se rían de mí, porque en la vida he visto esa enciclopedia de 1989 en CD-ROM, pero necesitaba ese argumento, como fuente de conocimiento, y sólo esa encajaba por fechas.

Admito que lo de obtener el conocimiento de una fuente externa, ¡es influencia del IBM Watson haciendo furor en el concurso Jeopardy!, y en algunos negocios. Sólo que Watson, lo consulta online de internet, y mi Jack, bueno, lo hacía en local como ahora decimos.

La enciclopedia de Compton/Britannica, condicionó que la historia se ambiente en Norteamérica, ya que forzaba a ser un país angloparlante, y el Reino Unido (como España), desde la muerte de los 8 bits, quedó retrasado.

Debo mucho de la inspiración a Rafael García González, el autor de Herbie, con el que he tenido contacto recientemente. Si en su época hubiera tenido los medios necesarios, mi Herbie, hubiera sido parecido a lo que aquí relato. Por cierto, que mi intento, que se quedó en sólo eso, un intento, se llamaba Dr. Jack, así que es también mi homenaje a él.

Puedo destacar la presencia del Rolex Milgauss 1019, protagonista inicial. No porque me guste especialmente, sino porque su resistencia a campos magnéticos lo hizo ser el reloj clave de los científicos y los técnicos. El Casio DW-5600C (Speed), en cambio sí que aparece por ser un reloj que me encanta. Habrás podido imaginar, que lo fácil hubiera sido que el propio programa Jack, en su versión de depuración, controlara el tiempo que tarda en responder. De hecho, se le llama corto-circuito, y se utiliza en programas que son no deterministas. Es decir, que pueden no acabar nunca, o no hacerlo en un tiempo razonable. Lo que se hace es que sea el programa el que cuente el tiempo, y digamos cuando lleve 2 segundos procesando, se quede con la mejor respuesta encontrada hasta el momento.

Quiero agradecer a Bianamaran, que, con su recopilatorio de relatos, me ha dado envidia, y por supuesto a José Antonio Carretero Sevilla, que, con La Conquista del Dragón, me ha motivado de nuevo.

El resto de la historia, destila mucho de mí, retroinformática, lenguaje C, compresión de datos, y bastante de historia de la tecnología. Me hubiera gustado ambientarla mejor en los años 80, en concreto en 1989, pero faltaban recursos, así que digamos que transcurre entre 1989 y 1990.

Como es habitual, barajé diferentes finales, e imagino que, para desgracia del lector, me quedé dejándolo abierto. La pregunta que deseo que surja es ¿Qué hubiera pasado si el Test de Turing se hubiera superado en 1990? O mejor aún… ¿Y si en realidad sí que se superó?

Finalmente, desconocía que, en 1990, el presidente de EE.UU. fuera George Bush padre. Me hubiera gustado que fuera Reagan, pero ambientarlo en esa época, me llevó inevitablemente a Bush.

La foto de la portada, es un Intel 80486DX, el corazón sobre el que late Jack.