



Universidade do Minho
Escola de Engenharia

Eduardo Jorge Costa Ribeiro

Etelvina
Um gerador de personagens para
Instant Messaging



Universidade do Minho

Escola de Engenharia

Eduardo Jorge Costa Ribeiro

Etelvina

Um gerador de personagens para
Instant Messaging

Dissertação de Mestrado
Mestrado em Informática

Trabalho efectuado sob a orientação do
Professor Doutor Alberto Manuel Brandão Simões

Setembro de 2010

É AUTORIZADA A REPRODUÇÃO PARCIAL DESTA TESE APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.

Universidade do Minho, ___ / ___ / _____

Assinatura: _____

Abstract

An Eliza is a computer program to simulate a chat, through natural language. This kind of computer program could have several functions, such as trying to maintain a chat with the user, offering some entertainment, or execute specific functions, with the goal of making the user interface more natural. This kind of approaches already has been used with commercial usage, from customer support to advertising.

This project was made with the wish of build an Eliza in the Portuguese language, fully configurable and expansible with modules, to be possible that any user could build is own Eliza character. Bearing in mind the increasing usage of instant messaging clients, social networking and blogs/microblogs, was choosen the creation of the Eliza based on this technologies to non recreational objectives, such as the answering of users questions about some subjects.

The development of this Eliza, which one is called Etelevina, was done by stages. In the first stage was developed a central module, which have as main functionalities the connection to the selected instant message server, and the coordination of the different modules that user choosed. The next stage included the elaboration of a language to configure or model the behaviour, that allows an easy way to construct modules.

To use as demonstration of the flexibility and the expansibility of this language, was build several modules, since simple modules of behaviour control (such as greetings or parting phrases), to complex modules (like the user question answering with informations acquired from Wikipedia processing).

Although have robustness problems, the implementation of the Etelevina structure was accomplished, being possible obtain an proof of concept of his flexibility and expansibility.

Resumo

Uma Eliza é um programa de computador que simula uma conversa através do uso da linguagem natural. Este tipo de programa pode ter diversas funções, tais como tentar manter uma conversa com o utilizador, oferecendo assim algum entretenimento, ou executar funcionalidades mais específicas, com o objectivo de tornar a interface com o utilizador mais natural. Este tipo de abordagens já tem vindo a ser utilizado com fins comerciais, desde o apoio a clientes à publicitação de produtos.

Com este trabalho pretende-se a construção de uma Eliza na língua portuguesa, totalmente configurável e expansível de forma modular, de modo a que qualquer utilizador possa construir a sua própria personagem Eliza. Dada a crescente utilização de clientes de mensagens instantâneas, as redes sociais e de blogues/microblogues, optou-se pela criação de uma Eliza baseada nestas tecnologias para fins não lúdicos, tais como a resposta a questões sobre determinados temas feitas sobre pelos utilizadores.

O desenvolvimento desta Eliza, a que chamamos Etelevina, foi realizada por fases. Inicialmente procedeu-se à construção de um módulo central, que tem como principais funções a ligação ao servidor de mensagens instantâneas configurado, e a coordenação dos diferentes módulos escolhidos pelo utilizador.

A fase seguinte compreendeu a elaboração de uma linguagem de configuração ou modelação de comportamento, que permite a construção de módulos de forma simples.

Para demonstrar a flexibilidade e expressividade desta linguagem foram montados vários módulos, desde simples módulos de modelação de comportamento (como saudações ou despedidas) até a módulos mais complexos (como a resposta a perguntas do utilizador com factos obtidos através do processamento da Wikipédia).

Embora com problemas de robustez, concluiu-se a implementação de toda a estrutura da Etelevina, obtendo-se uma prova de conceito da sua flexibilidade e expansibilidade.

Agradecimentos

Ao meu orientador Alberto Simões, pela paciência que teve, pela disponibilidade, pelo apoio na realização do projecto, pelas inúmeras ideias de funcionalidades para o projecto e também pelas opiniões que foram sempre ouvidas com agrado.

Ao professor José João Almeida, sempre bem disposto e constantemente a motivar com as suas ideias.

À Cristina, sempre disposta a ajudar e a motivar, pelo apoio na construção da ferramenta, testando sempre, pelo apoio na escrita deste documento, com constantes revisões, pela paciência, quando as coisas não corriam tão bem e a má disposição imperava, e pela insistência para que o projecto chegasse a bom porto.

Aos meus pais e à minha irmã, pois são sempre uma base de apoio para todas as situações.

Aos meus colegas, que sempre me apoiaram neste longo percurso.

A todos o meu obrigado!

Conteúdo

1	Introdução	1
1.1	Objectivos	2
1.2	Organização	3
2	Uma visão geral sobre <i>Elizas</i>	5
2.1	Uma visão histórica de <i>Elizas</i>	5
2.1.1	A linguagem de marcação AIML	11
2.2	Aplicação de <i>Elizas</i> no mundo real	14
2.2.1	<i>Elizas</i> de entretenimento e lazer	15
2.2.2	<i>Elizas</i> educacionais	16
2.2.3	<i>Elizas</i> de atendimento ao consumidor	18
3	Arquitectura da Etelvina	21
3.1	Estrutura de Implementação	21
3.1.1	Configurações a nível do motor da Etelvina	22
3.1.2	Configurações por módulo	29
3.1.3	Carregamento e utilização de módulos	41
3.1.4	Exemplos de módulos criados para demonstrar usabilidade da Etelvina	48
3.2	Conversas multi-utilizador	49
3.3	Estrutura de comunicações	50
3.3.1	Protocolo XMPP	50
3.3.2	API de comunicações utilizada - Net::XMPP	51
3.4	Etelvina <i>vs</i> AIML	52
4	Base de conhecimento da Etelvina	57
4.1	Algoritmos de obtenção de conhecimento	57
4.2	Construção de uma Base de Conhecimento	60
4.2.1	Utilização de conhecimento obtido na Etelvina	62
4.2.2	Infoboxes da Wikipédia	64
4.2.3	Exemplos de extracções de conhecimento de <i>infoboxes</i> elaboradas	68

4.2.4	Extracção de conhecimento de páginas genéricas da Wikipédia	73
5	Conclusões	83
5.1	Trabalho Futuro	84
	Referências Bibliográficas	87
	Anexos	91
A.	Conversa entre a <i>Eliza</i> e o <i>Parry</i> , na ICCC de 1972.	91
B.	Letra da música “ <i>Etelvina</i> ”	94

Lista de Tabelas

3.1	Exemplos de algumas transformações de código do ficheiro de configuração para código Perl.	35
3.2	Tabela que exemplifica as expressões regulares geradas para algumas frases de entrada.	36
3.3	Vantagens e desvantagens da utilização da metodologia utilizada para carregamento de módulos.	47
3.4	Tabela resumo de comparação entre a estrutura adoptada para a Etelvina e o AIML.	56

Lista de Figuras

2.1	Representação gráfica do jogo de imitação de Turing, no qual a pessoa C (operador), é incumbido de tentar determinar qual o participante - A ou B - é um computador e qual é um ser humano.	6
2.2	<i>Clippit</i> , o assistente do <i>Microsoft Office</i>	18
3.1	Representação da estrutura da Etelvina.	22
3.2	Estrutura de informações armazenadas por utilizador.	23
3.3	Schema de validação do XML de configuração do motor.	25
3.4	Esquema da estrutura de dados criada segundo o ficheiro de configuração representado no exemplo 3.2.	29
3.5	Ordem de processamento de módulos segundo a configuração.	30
3.6	Esquema representativo do fluxo de uma mensagem no módulo de meteorologia.	48
4.1	Diagrama que mostra o fluxo de aquisição de conhecimento perante um termo.	61
4.2	Exemplo de infobox da Wikipédia.	65
4.3	Exemplo de parte da infobox correspondente ao artigo sobre “Portugal”.	70

Lista de Exemplos

2.1	Categoria padrão em AIML.	12
2.2	Exemplo de configuração de categorias em AIML para " <i>knock-knock jokes</i> " incluído em "AIML Reference Manual" [RWTB10].	12
2.3	Exemplo da utilização da etiqueta <code><srail></code> para a configuração de várias perguntas para uma determinada resposta.	13
2.4	Exemplo da utilização da etiqueta <code><random></code> em AIML.	14
3.1	DTD de validação do XML de configuração do motor.	26
3.2	Exemplo de um ficheiro de configuração válido.	27
3.3	Exemplo de uma entrada de configuração por módulo.	31
3.4	Exemplo de criação de uma variável com o nome <i>num</i> e com o valor a ser um número inteiro positivo.	32
3.5	Exemplo de regras do tipo acção-reacção.	37
3.6	Exemplo da actualização do estado da Etelvina para um utilizador.	39
3.7	Exemplo de configuração de um módulo com respostas variáveis.	40
3.8	Exemplo de configuração de módulo após aplicação das regras de reescrita.	46
3.9	Utilização da API <code>Net::XMPP</code> para conexão ao servidor de mensagens instantâneas.	51
3.10	Utilização da API <code>Net::XMPP</code> para autenticação no servidor de mensagens instantâneas e tratamento de eventos de mensagens e presença.	51
3.11	Exemplo da utilização da etiqueta <code><system></code> em AIML, utilizando os comandos <i>echo</i> e <i>date +%H</i> disponível num ambiente UNIX, e que devolve a hora actual.	52
3.12	Exemplo da utilização da etiqueta <code><system></code> em AIML, em que é necessário efectuar um pré-processamento, para obter o nome da personagem construída.	53
3.13	Exemplo da utilização da etiqueta <code><javascript></code> em AIML para o cálculo da raiz quadrada de um número.	53
3.14	Exemplo da configuração da raiz quadrada de um número na Etelvina.	54
3.15	Exemplo da utilização da etiqueta <code><srail></code> e <code><random></code> em AIML.	54
3.16	Exemplo de utilização, na Etelvina, de várias mensagens de entrada e saída numa só regra.	55
4.1	Exemplo do código da infobox correspondente à figura 4.2	65

- 4.2 Exemplo de código de campos variáveis no artigo “Portugal”. 66
- 4.3 Extracto de código do artigo "Lista de países e estados soberanos em 2009"[Wik10a]. 71
- 4.4 Extracto de código do artigo "Lista de freguesias de Portugal"[Wik10b]. 72

Lista de Conversas

2.1	Exemplo de uma conversa entre um utilizador e a Eliza descrita por Weizenbaum no seu artigo [Wei66]	6
2.2	Exemplo de uma conversa entre um entrevistador e o <i>Parry</i> descrita por Bertram Raphael no seu livro [Rap76], que demonstra alguns traços da sua personalidade.	8
2.3	Excerto da conversa entre a <i>Eliza</i> e o <i>Parry</i> , na ICCC de 1972. Conversa completa no Anexo A.	10
2.4	Exemplo de uma conversa entre um entrevistador e Julia.	10
2.5	Exemplo de conversa de acordo com a configuração do exemplo 2.2.	13
2.6	Transcrição de uma conversa entre um jurado do prémio Loebner 2009 e o <i>Do-Much-More</i>	15
2.7	Exemplo de uma conversa com o <i>Talk to FRANK</i>	16
2.8	Exemplo de utilização do <i>Buscas</i>	19
3.1	Exemplo de uma conversa com a Etelvina, perante a configuração definida no exemplo 3.3.	31
3.2	Exemplo de uma conversa com a Etelvina utilizando as expressões matemáticas definidas na gramática 3.1.	34
3.3	Exemplo de uma conversa através de regras de acção-reacção.	37
3.4	Exemplo de uma conversa utilizando as regras definidas, com indicações temporais, no exemplo 3.7.	40
4.1	Exemplo de conversa com informações de países, obtidos através da extracção de informação de infoboxes da Wikipédia.	71
4.2	Exemplo de uma resposta da Etelvina em que o resultado não é o pretendido.	75
4.3	Outro exemplo de uma resposta da Etelvina em que o resultado não é o pretendido.	76
4.4	Exemplo de um resultado obtido conforme o esperado, dado a especificidade do termo na mensagem de entrada.	76
4.5	Exemplo de respostas devolvidas pela Etelvina, após a implementação da percentagem de relevância.	76
4.6	Exemplo de pesquisas na Wikipédia, em páginas genéricas, através da Etelvina.	77

4.7 Exemplo de uma resposta da Etelvina com a listagem de resultados de artigos ambíguos.	80
--	----

Lista de Acrónimos e Siglas

AIML	Artificial Intelligence Markup Language
ALICE	Artificial Linguistic Internet Computer Entity
API	Application Programming Interface
DTD	Document Type Definition
IE	Information extraction
IM	Instant Messaging
MSNP	Microsoft Notification Protocol
NHS	National Health Service
QA	Question Answering
TCP	Transmission Control Protocol
TLS	Transport Layer Security
VoIP	Voice over Internet Protocol
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

Capítulo 1

Introdução

A comunicação é uma necessidade que é inata ao Homem. Desde cedo que a comunicação surgiu, não da forma moderna que hoje se utiliza, mas com formas diferentes, mais simples, tais como a utilização de gestos e imagens. A comunicação não é só a utilização da língua para transmissão de informação mas sim qualquer processo em que se codifica e transmite informação a partir de um emissor, para um receptor, através de um canal.

A linguagem humana moderna evoluiu, exigindo o desenvolvimento do trato vocal utilizado para a produção da fala e das habilidades cognitivas necessárias para produzir expressões linguísticas. Assim, e dada a necessidade de comunicar, foram várias as invenções que permitiram ao Homem ultrapassar barreiras como a distância, tais como o telégrafo, o telefone, passando também pela imprensa, rádios, televisões, até a uma das formas com mais poder de comunicação nos dias de hoje: a *Internet*.

Com a evolução da *Internet*, as pessoas têm vindo a encontrar novas formas para comunicarem, desde o, mais antigo, correio electrónico, até à nova moda das redes sociais como *MySpace* e o *Facebook*, passando pelos serviços de mensagens instantâneas como o *Microsoft Messenger* e o *Google Talk*, de fóruns e até mesmo por *blogs* ou *Twitter*, seja através de computadores ou telemóveis. Todos estes serviços têm um mesmo objectivo: comunicar. Essa comunicação, como qualquer outra, desde os primórdios da civilização, pode surgir apenas por lazer, de forma a conviver com outras pessoas, até à comunicação por motivos profissionais, de forma a se trabalhar cooperativamente.

A possibilidade de, em qualquer local, se conseguir aceder a serviços na *Internet*, tornando possível a comunicação com outros utilizadores, levantou novos hábitos nas comunidades, associados à constante necessidade de informação. Os utilizadores de tais serviços começam a tentar obter informação através de todos os seus contactos/seguidores de cada um dos serviços online, evitando assim as pesquisas mais comuns.

A necessidade de comunicação levou à criação, há alguns anos atrás, no ano

de 1966, de um programa de computador que tinha como objectivo simular um interlocutor a ter uma conversa com um ser humano. Esse programa tinha o nome de *Eliza*, e é ainda por esse nome que são vulgarmente denominados todos os programas que simulam diálogos. A *Eliza* original foi criada com o objectivo de passar o teste de Turing, que é um teste criado nos anos 50 do século XX, por Alan Turing, para testar a inteligência artificial de um computador, denominado pelo autor como o 'Jogo de imitação'¹. O teste consiste em saber se um operador consegue ou não diferenciar se as respostas a perguntas elaboradas por ele foram respondidas ou não por um computador. Se o operador não consegue diferenciar qual o ser humano e qual o computador, o programa passa o referido teste[Tur50].

Utilizando o conceito de *Eliza*, com uma interface através de um contacto de um servidor de mensagens instantâneas, surgiram os objectivos de transformar uma *Eliza* num contacto “amigo” de um cliente de mensagens instantâneas, de forma a obter respostas a variados tipos de questões, desde as mais simples até às mais complexas, assim como obter lazer e diversão com esse mesmo contacto. Embora não se queira uma *Eliza*, pretende-se que a Etelvina tenha uma camuflagem real, adicionando-lhe uma biografia, de forma a que a Etelvina tenha a capacidade de dar resposta a perguntas relativamente comuns sobre si própria, tal como o nome, data de nascimento e outros dados do género.

Assim sendo, e como reflexo das necessidades da sociedade referidas, surgiu o projecto *Etelvina - Um gerador de personagens para Instant Messaging*. A Etelvina é uma Eliza, totalmente na língua portuguesa, que tem a capacidade de ser expansível e configurável, de forma que o utilizador, através da construção de novos módulos ou simplesmente de regras para padrões por si definidos, obtenha o comportamento desejado.

Dada a utilização de clientes de mensagens instantâneas para os mais variados tipos de comunicação e necessidades de obtenção de informação, faz sentido que se agilize o uso de diferentes programas cobrindo-os como uma personagem, ou um ser de silício, que seja a personificação de todos esses programas ou serviços.

1.1 Objectivos

Tendo como principal motivação para a construção deste projecto a crescente utilização de serviços de comunicação através da *Internet*, tais como clientes de mensagens instantâneas, assumiram-se como objectivos os seguintes pontos:

- i. **Tornar a Etelvina uma presença funcional, sempre disponível num cliente de Instant Messaging.**

Pretende-se que a Etelvina funcione como um “amigo”, sempre presente, que seja capaz de fornecer informações úteis, tais como o estado meteorológico em de-

¹Do inglês *Imitation Game*

terminado local, o significado de determinada palavra, fazer cálculos aritméticos simples, a consulta de um motor de pesquisa online, entre outras possibilidades de um conjunto variado de utilitários.

ii. Construir uma ferramenta configurável, flexível e expansível, em que seja fácil a um qualquer utilizador compor módulos obtendo o comportamento desejado.

Propõe-se a criação de uma Eliza diferente, com a capacidade de ser flexível e expansível, de forma que seja possível alterar o protocolo de comunicação que utiliza para suportar uma ferramenta de mensagens instantâneas diferente, assim como implementar novas funcionalidades, construindo uma personalidade diferente.

Não se pretende criar uma Eliza comum com o objectivo de passar o teste de Turing[Tur50], teste este descrito mais detalhadamente na secção 2.1, mas sim uma plataforma de geração de personagens para *Instant Messaging* com objectivos mais funcionais do que lúdicos.

Entenda-se por gerador de personagens de *Instant Messaging* a capacidade de, alterando somente pequenos ficheiros de configuração, alterar completamente a personagem criada. Por isso, foi necessário desenvolver um motor sólido sobre o qual se possa implementar uma linguagem de padrões para configuração do comportamento da Etelvina.

No âmbito da construção de uma Eliza configurável, flexível e expansível, outro dos objectivos passa por poder construir uma Eliza segundo padrões/regras e estados, de forma a se conseguir, através desses mesmos padrões, ter a capacidade de dar uma resposta simples no seguimento de uma conversa, assim como adicionar, remover e alterar as regras. Embora não se pretenda uma Eliza, pretende-se dar uma camuflagem real à Etelvina, adicionando-lhe biografia, entre outras informações, de forma que se consiga a obtenção de uma Eliza com alguns comportamentos humanos. No entanto, apesar de conseguir ter conversações simples, a Etelvina não deve tentar responder como um ser humano, pois assim perde parte da sua componente funcional.

iii. Permitir o desenvolvimento de módulos com conhecimento próprio.

Por último, a Etelvina deverá ter mecanismos que a possibilitem armazenar conhecimento de forma a conseguir responder a perguntas.

1.2 Organização

A dissertação encontra-se estruturada em seis capítulos:

Capítulo 1 - Capítulo onde é feita uma apresentação do problema em questão, assim como os objectivos propostos.

Capítulo 2 - Capítulo onde se faz uma referência ao estado da arte do desenvolvimento de Elizas, dando uma perspectiva histórica e alguns exemplos de Elizas que deixaram a sua marca.

Capítulo 3 - Capítulo em que são explicadas quais as estruturas de comunicação e de desenvolvimento da Etelvina, assim como os métodos utilizados para configuração de comportamentos.

Capítulo 4 - Capítulo onde é referida a forma como é extraída informação para a base de conhecimento da Etelvina.

Capítulo 5 - Capítulo onde se refere as conclusões obtidas, assim como uma pequena descrição de trabalho futuro.

Capítulo 2

Uma visão geral sobre *Elizas*

Uma *Eliza* é um programa de computador que, através da linguagem natural, tem a capacidade de ter um diálogo com um utilizador.

Os objectivos das *Elizas* são, na maioria dos casos, conseguir enganar um utilizador que está a ter a conversa com a *Eliza*, de forma a que esse utilizador pense que quem está do outro lado na conversa seja um ser humano.

Com a vulgarização da utilização da Internet através da World Wide Web, as *Elizas* têm sido utilizadas nas mais variadas tarefas, tais como o mero entretenimento, mas também em funções de ensino à distância em plataformas electrónicas (o *e-learning*), e de atendimento ao consumidor.

Neste capítulo faz-se uma visão geral sobre o conceito *Eliza*, a sua evolução, e as suas aplicações.

2.1 Uma visão histórica de *Elizas*

Em 1950, Alan Turing, no seu livro *Computing Machinery and Intelligence*[Tur50] propôs o jogo de imitação como um substituto para a questão “Podem as máquinas pensar?”. Ele previu que em 50 anos, ou seja, até ao ano 2000, o progresso tecnológico deveria produzir máquinas de computação com memória de 10^9 bits, e que com essa máquina, um programa de computador seria capaz de enganar um interlocutor comum por 5 minutos, isto em cerca de 30% das vezes [Tur50].

Em 1966, no *MIT Artificial Intelligence Lab*, Joseph Weizenbaum criou o primeiro programa com o objectivo de passar o teste de Turing, apelidado de *Eliza*. Este programa, escrito em apenas 204 linhas de código, começou a história das aplicações agora denominadas tradicionalmente por *Elizas*.

A *Eliza* tinha como base uma personagem que simulava uma psicoterapeuta, construída com base no princípio psicanalítico Rogeriano¹. Este princípio da psico-

¹Do nome do seu criador, Carl Rogers, um influente psicólogo americano, o principal impulsor e maior responsável pelo desenvolvimento da Psicologia Humanista.

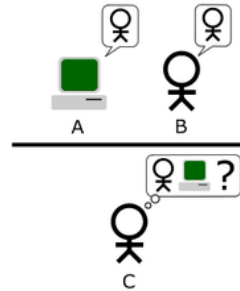


Figura 2.1: Representação gráfica do jogo de imitação de Turing, no qual a pessoa C (operador), é incumbido de tentar determinar qual o participante - A ou B - é um computador e qual é um ser humano.

logia consiste num conceito chave, em que a pessoa que melhor pode compreender e mudar o cliente² é ele próprio. A tarefa do terapeuta consiste na criação das condições nas quais o cliente se sinta suficientemente livre e suficientemente forte para se ajudar a si próprio. O próprio Carl Rogers tem a seguinte afirmação: “É o próprio cliente que sabe aquilo de que sofre, em que direcção se deve ir, quais os problemas que são cruciais, que experiências foram profundamente recalçadas.”[Rog61].

De forma a conseguir que a *Eliza* representasse uma psicoterapeuta, Joseph Weizenbaum criou uma metodologia para a *Eliza* seguir a sua conversa, que consiste na repetição das frases do cliente, de modo a que consiga que o cliente faça uma introspeção, sem ser necessário o envolvimento de opiniões do psicoterapeuta, neste caso, da *Eliza*. Este género de conversa foi escolhido pois, segundo Weizenbaum, “um dos dois participantes é livre para assumir a pose de não conhecer quase nada do mundo real”[Wei66].

Deste modo a *Eliza* consegue esconder as fragilidades que uma máquina tem quando tenta simular um ser humano numa conversação, dando a iniciativa da conversa a quem fale com ela, com frases como “Por favor conte mais” quando não consegue compreender sobre o que o utilizador está a falar, ou “Está a ser um pouco negativo” quando existe uma palavra conotada com a negatividade, conseguindo assim dar sempre uma resposta a qualquer que fosse a conversa.

Conversa 2.1: Exemplo de uma conversa entre um utilizador e a *Eliza* descrita por Weizenbaum no seu artigo [Wei66]

Eliza: Hi, my name is Eliza, what is your name?
Utilizador: My name is user, how are you?
Eliza: I am fine
Utilizador: men are all alike.
Eliza: in what way?
Utilizador: they're always bugging us about something or other.
Eliza: can you think of a specific example?

²Termo utilizado na área da Saúde para descrever o doente/paciente

Utilizador: well, my boyfriend made me come here.
Eliza: your boyfriend made you come here?
Utilizador: he says i'm depressed much of the time.
Eliza: i am sorry to hear you are depressed.
Utilizador: it's true. i am unhappy
Eliza: do you think coming here will help you not to be unhappy?
Utilizador: i need some help, that much seems certain.
Eliza: what would it mean to you if you got some help?
Utilizador: perhaps i could learn to get along with my mother.
Eliza: tell me more about your family.
Utilizador: my mother takes care of me.
Eliza: who else in your family takes care of you?
Utilizador: my father.
Eliza: your father?
Utilizador: you are like my father in some ways.
Eliza: what resemblance do you see?
Utilizador: you are not very aggressive but i think you don't want me to notice that.
Eliza: what makes you think i am not very aggressive?
Utilizador: you don't argue with me.
Eliza: why do you think i don't argue with you?
Utilizador: you are afraid of me.
Eliza: does it please you to believe i am afraid of you?
Utilizador: my father is afraid of everybody.
Eliza: what else comes to your mind when you think of your father?
Utilizador: bullies. ok, bye bye.
Eliza: bye

Uma das características mais marcantes da *Eliza* é a forma como utiliza as frases dos utilizadores na sua conversa. Weizenbaum explica, no seu artigo [Wei66] a forma como é efectuada essa utilização. Em frases como “*I am BLAH*”, Weizenbaum afirma que se pode dar uma resposta do género “*How long have you been BLAH*”, independentemente do significado de *BLAH*. Outro exemplo mais complicado é uma frase do tipo “*It seems that you hate me*”. Neste caso, a *Eliza* apenas utiliza a frase a partir do sujeito e faz a substituição de *you* para *I* e de *me* para *you* e adiciona uma frase tipo no início da resposta, como a frase “*What makes you think*” e o ponto de interrogação no final. Assim a resposta devolvida seria “*What makes you think I hate you?*”, o que faz com que a conversa se assemelhe a uma conversa humana. No entanto a técnica é falível com frases do género “*I am doing fine thank you*”. Neste caso, em que o utilizador diz à *Eliza* que está bem, agradecendo no final, a *Eliza* devolverá, utilizando a técnica supracitada, a resposta “*How long have you been doing fine thank I?*” [Lav10], frase esta que não faz sentido. Esta resposta reforça a ideia de que a *Eliza* reage de acordo com um sistema de acções baseadas em padrões, que, ao detectar um padrão que reconhece, devolve automaticamente a resposta que tem definida.

Apesar da *Eliza* ter um disfarce, que consiste na representação de um psicoterapeuta, repetindo as frases do utilizador para que este continue a conversa, tal como

supracitado, que a ajudava a tornar-se mais inteligente e humana aos olhos de quem a interpelasse, não foi o suficiente para conseguir passar o *Teste de Turing*, teste que até à data nunca foi ultrapassado apesar de ter sido formulado há mais de 50 anos. No entanto, o desenvolvimento da *Eliza* despertou o interesse de diversos investigadores, tendo ficado para a história o trabalho desenvolvido por Kenneth Colby, psiquiatra de profissão, com o *Parry*.

O *Parry* é uma *Eliza* que simula o comportamento de um paranóico esquizofrénico³, e foi implementado sobre um modelo baseado em conceitos, conceptualizações e crenças. *Parry* foi pensado como um sistema de ensino virtual para estudantes de psiquiatria antes de iniciarem um contacto com pacientes reais[Bod08].

Colby chegou mesmo a sujeitar *Parry* a testes médicos “às cegas”, onde os médicos iam questionando tanto o programa, como três pacientes humanos diagnosticados como paranóicos. As análises das conversas de psiquiatras e cientistas informáticos revelaram que nenhum destes grupos de pessoas conseguiu distinguir o programa de computador dos pacientes paranóicos reais.

As opiniões desvalorizaram por diversas vezes o *Parry*, pois o modelo de escolher respostas aleatórias não modelava um paciente humano. No entanto, o *Parry*, ao contrário da *Eliza*, tinha uma personalidade [Mau94], o que significava uma evolução. Isto porque enquanto a *Eliza* apenas fazia algumas questões, assim como a repetição de frases do seu paciente, de modo a que não fosse necessário fornecer as suas opiniões, enquanto que o *Parry*, apesar de simular um comportamento paranóico, com frases que por vezes não faziam muito sentido, possuía uma personalidade, baseada em conceitos, crenças e pequenas histórias.

Conversa 2.2: Exemplo de uma conversa entre um entrevistador e o *Parry* descrita por Bertram Raphael no seu livro [Rap76], que demonstra alguns traços da sua personalidade.

Entrevistador: How are you today?
Parry: OK.
Entrevistador: What is your name?
Parry: My name is Frank Smith.
Entrevistador: How old are you?
Parry: Twenty-eight.
Entrevistador: Are you married?
Parry: No.
Entrevistador: What sort of work do you do?
Parry: I work in the Post Office.
Entrevistador: Why are you in the hospital?
Parry: I shouldn't be here.
Entrevistador: Who brought you here?
Parry: The police.
Entrevistador: What trouble did you have with the police?
Parry: Cops don't do their jobs.
Entrevistador: What do they do instead?

³*Parry* trata-se de um doente paranóico, que é um cliente da *Eliza*, uma psicoterapeuta.

Parry: Cops arrest the wrong people.
Entrevistador: They arrest the right ones sometimes.
Parry: The law should get those Italian crooks.
Entrevistador: Are you Italian?
Parry: My background is British, but I was born in this country.
Entrevistador: Do you know any Italian crooks?
Parry: I try to avoid the underworld.
Entrevistador: Are you afraid of the underworld?
Parry: The mafia are the worst gangsters.
Entrevistador: Are you connected with the mafia?
Parry: You know, they know me.
Entrevistador: How do they know you?
Parry: They spy on me.
Entrevistador: Why do they?
Parry: They want to get false evidence to frame me.
Entrevistador: What do they have against you?
Parry: They are out to get me.
Entrevistador: What for?
Parry: I have incriminating evidence against the mafia.
Entrevistador: Are you sure of all this?
Parry: You don't believe me?

Tal como a *Eliza* possuía os seus truques, como a reutilização de uma frase preferida pelo utilizador, o *Parry* possuía ainda mais [Col75]. Esses truques consistem na utilização de respostas tais como “Não sei” a uma pergunta, na tentativa de fuga ao tema da conversa com respostas “Por que é que perguntas isso?”, alterando assim o tema, e também a inclusão de pequenas histórias, tais como histórias sobre a máfia. Estes truques, que simulam bem o comportamento paranóico, limitam a extensão da conversa e, quando repetidos em demasia, acabam por desmascarar o interlocutor. No entanto, e tal como foi supracitado, Kenneth Colby conseguiu convencer um grupo de psiquiatras a entrevistarem o *Parry* por teleescritor⁴. Após a entrevista a *Parry*, Colby pegou no documento com a entrevista juntou a outras entrevistas a humanos paranóicos, e entregou a outro grupo de psiquiatras, dizendo-lhes “Uma destas conversas foi com um computador. Conseguem descobrir qual foi?”. Nenhum deles conseguiu distinguir a conversa que tinha sido criada com o *Parry*[Teu04].

A *Eliza*, programa que simulava um psicoterapeuta, e o *Parry*, que simulava um doente paranóico esquizofrénico, “encontraram-se” por diversas vezes, tendo o mais famoso encontro entre os dois ocorrido na International Conference on Computer Communication (ICCC), em Outubro de 1972, na cidade de Washington DC.

⁴Aparelho, inventado em 1897, que, aplicado a uma rede telegráfica ou telefónica, põe em movimento um mecanismo que imprime palavras à distância.

Conversa 2.3: Excerto da conversa entre a *Eliza* e o *Parry*, na ICCC de 1972. Conversa completa no Anexo A.

Parry: Do you know anything about bookies?
Eliza: What makes you think I know anything about bookies?
Parry: A bookie didn't pay me off once.
Eliza: What does that suggest to you?
Parry: I beat up the bookie who did not pay off, and I thought he might use his friends in the underworld to get even with me.
Eliza: What else would you like to discuss?
Parry: What do you know about racketeers?
Eliza: Why do you ask?
Parry: What experience have you had gambling?
Eliza: Why does that question interest you?
Parry: I like horseraces.
Eliza: What else do you like?
Parry: I know the mob controls the big rackets.
Eliza: Who are you really talking about?
Parry: Have you ever really thought about the MAFIA?
Eliza: Suppose you ever really thought about the MAFIA.
Parry: Let's talk about something else.
Eliza: I am not sure I understand you fully.
Parry: You keep asking me the same things, why not change the topic?
Eliza: In what way?
Parry: What do you mean?
Eliza: Why are such questions on your mind?
Parry: I thought I told you, people get on my nerves.

A partir do início da década de 1990, apesar da popularidade de sistemas que se baseavam na *Eliza*, os cientistas de computação, insatisfeitos com os resultados obtidos até então com os sistemas com base em técnicas de *match* de padrões, iniciaram o desenvolvimento de *Elizas* com técnicas mais sofisticadas, do ponto de vista da Inteligência Artificial, tais como as redes neuronais.

Uma *Eliza* desse género que conseguiu bons resultados é a *Julia*, escrita por Michael Mauldin, no *Center for Machine Translation da Carnegie-Mellon University*.

A *Julia* destacou-se por ser um agente neuronal que habita em ambientes de realidade virtual baseados em texto, fazendo exploração autónoma do ambiente habitado por ela.

Além de conversar com o utilizador, tem também a capacidade de recordar informações referentes a outros utilizadores e as acções que efectuou anteriormente.

A conversa 2.4 seguinte é um extracto de um diálogo com *Julia* no Prémio Loebner.

Conversa 2.4: Exemplo de uma conversa entre um entrevistador e *Julia*.

Utilizador: have you ever kept a penthouse pet?

Julia: do you have any pets?
Utilizador: i think you're avoiding my question.
Julia: do you think animals can think?
Utilizador: do you think it ethical for one species enslave another?
Julia: what kinds of pets did you have as a child?
Utilizador: i'm asking the questions. what is a pet if it is not a slave of speciesism.
Julia: oh, that's so sad. i think pets enrich a childhood immensely.
Utilizador: how can we keep some animals as pets, yet eat others?
Julia: guinea pigs are as smart as cucumbers.
Utilizador: would you eat a guinea pig?
Julia: i spent several summers on my uncle's farm. i really enjoyed the animals a lot.

O Prémio Loebner foi criado em 1990, quando Hugh Loebner concordou com o centro de Cambridge de Estudos Comportamentais para a criação de uma competição projectada com o intuito de ultrapassar o teste de Turing. Loebner prometeu um prémio de \$100.000 e uma medalha de ouro para o primeiro computador em que as respostas fossem indistinguíveis de um ser humano, ou seja, que passe o Teste de Turing.

Todos os anos é atribuído um prémio anual de \$2000 e uma medalha de bronze à *Eliza* que mais se assemelhe com um ser humano. Assim, através destes prémios, Loebner e o centro de Cambridge de Estudos Comportamentais conseguiram que surgissem novas *Elizas*, tal como o projecto vencedor nos anos de 2000, 2001 e 2004. Apesar de ter ganho o prémio Loebner por três vezes, este projecto não foi capaz de passar o teste de Turing.

No Prémio Loebner de 2000, Richard Wallace apresentou um sistema inovador, chamado *ALICE* (*Artificial Linguistic Internet Computer Entity*) [Wal00].

A maior novidade de *ALICE* foi a utilização de *AIML* (*Artificial Intelligence Markup Language*), uma linguagem de marcação baseada em *XML*⁵ (*eXtensible Markup Language*) para representar a sua base de conhecimento.

Dada a sua vertente baseada em padrões, comparável à linguagem de domínio específico apresentada nesta dissertação, é apresentada na próxima secção uma discussão mais detalhada das potencialidades da *AIML*.

2.1.1 A linguagem de marcação AIML

Wallace definiu a *AIML* (*Artificial Intelligence Markup Language*) como uma linguagem de marcação para *Elizas* baseada em unidades denominadas “categorias” que podem ser comparadas ao raciocínio baseado em casos [Kol93]. Uma categoria em *AIML* é estruturada da seguinte forma:

⁵<http://www.w3.org/XML/>

Exemplo 2.1: Categoria padrão em AIML.

```
<category>
  <pattern> padrão de entrada </pattern>
  <template> molde para construção de respostas </template>
</category>
```

A unidade básica de conhecimento em AIML é a “categoria”. Cada categoria consiste numa questão ou padrão de entrada, uma resposta de saída e, opcionalmente, um contexto. A questão é denominada de “pattern”. A resposta tem o nome de “template”. Os dois tipos de contexto opcionais são chamados de “that” e “topic”. A linguagem AIML é simples, consistindo apenas em palavras, espaços, o símbolo sublinhado (_)⁶ e o asterisco (*)⁷. As palavras podem ser compostas de letras e números, mas não de outros caracteres. O uso de maiúsculas na linguagem AIML é irrelevante. As palavras são separadas por um espaço único, e os símbolos especiais já referidos funcionam como palavras.

De forma a tornar esta linguagem mais dinâmica, em AIML existem dois modos de executar programas. A etiqueta **<system>** executa qualquer programa que esteja acessível na linha de comandos do sistema operativo e insere os resultados na resposta. A segunda abordagem é baseada na etiqueta **<javascript>**, que permite a inserção de código em *javascript* nos templates, mas que apenas são executados caso *Eliza* for executada num browser.

Exemplo 2.2: Exemplo de configuração de categorias em AIML para “*knock-knock jokes*” incluído em “AIML Reference Manual” [RWTB10].

```
<category>
  <pattern>KNOCK KNOCK</pattern>
  <template>Who's there?</template>
</category>
<category>
  <pattern>*</pattern>
  <that>WHO IS THERE</that>
  <template><person/> Who?</template>
</category>
<category>
  <pattern>*</pattern>
  <that>* WHO</that>
  <template>Ha ha very funny, <getname/></template>
</category>
```

⁶O símbolo “_” faz match com qualquer texto, sendo o primeiro símbolo a ser analisado pelo interpretador AIML, pois é o símbolo com maior prioridade no *matching*. Ou seja, faz match com a maior quantidade possível de texto.

⁷O símbolo “*” faz match com qualquer texto, no entanto, ao contrário do símbolo “_”, é o símbolo com mais baixa prioridade no *matching*. Faz match à menor quantidade possível de texto.

Uma das etiquetas mais utilizadas nesta linguagem é a `<that>`. Pode aparecer dentro das categorias, e deve conter um padrão. Só se o seu padrão fizer *match* à última expressão respondida pela *Eliza* é que a acção é despoletada. Um exemplo muito interessante de uma aplicação para esta etiqueta na AIML é descrito em “AIML Reference Manual” [RWTB10], nas tradicionais piadas na língua inglesa denominadas de “*knock-knock jokes*”, tal como descrito no exemplo 2.2 .

Conversa 2.5: Exemplo de conversa de acordo com a configuração do exemplo 2.2.

```

Utilizador: Knock knock.
Eliza: Who's there?
Utilizador: Banana.
Eliza: Banana who?
Utilizador: Knock knock.
Eliza: Who's there?
Utilizador: Banana.
Eliza: Banana who?
Utilizador: Knock knock.
Eliza: Who's there?
Utilizador: Orange.
Eliza: Orange who?
Utilizador: Orange you glad I didn't say banana.
Eliza: Ha ha very funny, Utilizador.

```

No exemplo 2.2, é visível que a etiqueta `<template>` é a frase “Who’s there?”, e na regra seguinte a etiqueta `<that>` é “WHO IS THERE”. Em AIML, apesar de serem feitas transformações, tais como remoção de acentos, símbolos de pontuação e normalização de espaços, não são válidas para transformações de palavras como “Who’s” e “WHO IS”. Assim, para o sistema reconhecer ambas como palavras homónimas, é necessário criar novas categorias para permitir tal situação.

De forma a permitir o reaproveitamento de respostas já definidas, é possível indicar como resposta uma pergunta equivalente, onde a resposta oficial deve ser consultada. Para isso é usada a etiqueta `<srail>` (ver exemplo 2.3).

Exemplo 2.3: Exemplo da utilização da etiqueta `<srail>` para a configuração de várias perguntas para uma determinada resposta.

```

<category >
  <pattern>OLA</pattern >
  <template>Olá!</template >
</category >
<category >
  <pattern>OI</pattern >

```

⁷Em AIML, a etiqueta `<person/>` corresponde a `<person><star/></person>`, e é uma etiqueta onde o interpretador de AIML muda o pronome pessoal numa frase.

```

    <template>
      <srai>OLA</srai>
    </template>
</category>
<category>
  <pattern>BOM DIA</pattern>
  <template>
    <srai>OLA</srai>
  </template>
</category>

```

Como é possível verificar no exemplo 2.3, a mesma resposta é usada para todas as questões configuradas, neste caso as mensagens “Ola”, “Oi” e “Bom Dia”. Assim, apenas é necessário configurar uma resposta para as mensagens de entrada com o mesmo significado. Esta etiqueta é utilizada também para fazer substituições de abreviaturas e de contracções de palavras.

De forma a se obter o comportamento inverso, isto é, de se ter apenas uma questão e múltiplas respostas para essa questão, existe a etiqueta `<random>`, onde se colocará uma lista de respostas possíveis com a etiqueta `` (ver exemplo 2.4).

Exemplo 2.4: Exemplo da utilização da etiqueta `<random>` em AIML.

```

<category>
  <pattern>*</pattern>
  <template>
    <random>
      <li>What’s your favorite car?</li>
      <li>What kind of car do you drive?</li>
      <li>Do you get a lot of parking tickets?</li>
      <li>My favorite car is one with a driver.</li>
    </random>
  </template>
</category>

```

Mais informação sobre AIML, assim como sobre o projecto ALICE, pode ser encontrado em <http://www.alicebot.org>.

2.2 Aplicação de *Elizas* no mundo real

As aplicações das *Elizas* no mundo real são variadas, desde o simples entretenimento ou lazer, até ao ensino à distância, ao atendimento ao consumidor ou mesmo até *Elizas* maliciosas.

2.2.1 *Elizas* de entretenimento e lazer

Tendo em conta o objectivo inicial das *Elizas*, que consiste na simulação de conversas entre humanos, este tipo de aplicação no mundo real é o mais antigo. Para que consiga divertir o utilizador, as *Elizas* necessitam ter um comportamento que não seja sempre igual, de forma a simular uma personalidade ou de abstrair o utilizador da falta dela, como acontecia com a *Eliza* de Joseph Weizenbaum.

Além de ser a aplicação mais antiga, pode dizer-se também que é a mais comum, dado o grande número de *Elizas* que existem com esta finalidade.

Conversa 2.6: Transcrição de uma conversa entre um jurado do prémio Loebner 2009 e o *Do-Much-More*.

Jurado: Hi, how is it going?
Do-Much-More: Hello. How are you doing?
Jurado: Good, who are you?
Do-Much-More: Well, I am studying engineering. That ought to give you an idea about who I am.
Jurado: No, not really?
Do-Much-More: I have a terrific yearning to know something more about experiments?
Jurado: Where are you from?
Do-Much-More: I don't know, but don't forget I can't be in three places at once.
Jurado: What lang city do you live in?
Do-Much-More: What city – that's rather a tricky question. The trouble is, one city seems to be much like another.
Jurado: Not really. Where do you live?
Do-Much-More: In the old dairy next to the river, I believe.
Jurado: Interesting, so you are a farmer then?
Do-Much-More: Well, if you ask a physiologist what I am, he won't say I'm a farmer. He'll say I consist mostly of water – enough to fill a large beer tun – plus some iron, phosphorus, and lots of other prosaic ingredients.
Jurado: Wie eird answer to a simple question, wouldn't you say?
Do-Much-More: I certainly would. Just watch me!
Jurado: What will I see?
Do-Much-More: You don't have to worry about what you see, you shouldn't see it on a cold winter's evening.
Jurado: What do you make of the Arctic Monkeys?
Do-Much-More: Well, here's a clue: I make what a keeper in a zoo would make. Try that for size!

Bons exemplos deste tipo de aplicação são as *Elizas* que foram sendo descritas na secção anterior, bem como o *Elbot* [Rob10], o *Jabberwacky* [Car10], o *Ultra Hal* [Inc10], o *Do-Much-More* [Lev10], vencedor do prémio Loebner de 2009, ou até mesmo um que é capaz de falar português, o *Senhor Bot*, criado pela Microsoft, utilizando o cliente de mensagens instantâneas *Windows Live Messenger*.

2.2.2 *Elizas* educacionais

O crescimento das tecnologias de informação e comunicação permitiram que técnicas de ensino à distância, vulgarmente denominadas de *e-learning*, fossem utilizadas para que as pessoas aprendam sem necessidade de se deslocarem.

Apesar de todas as vantagens que possui este tipo de ensino, uma vez que não existe contacto presencial com outros seres-humanos, tende a ser desmotivador pois só existe o contacto homem-máquina. As *Elizas* podem tornar-se assistentes virtuais de aprendizagem, diminuindo o nível de isolamento dos utilizadores deste tipo de metodologia de ensino.

As *Elizas ADELE* [JS97] e *STEVE* [RJ97] são exemplos de como este tipo de aplicações tem sido utilizado como plataforma de *e-learning*. Têm como principais funcionalidades a monitorização das acções dos estudantes, a resposta às suas perguntas e a demonstração de como resolver determinado problema, no contexto do tema ensinado.

Outra das áreas onde as *Elizas* conseguem obter resultados interessantes é no ensino de novas línguas. São várias as *Elizas* que têm este objectivo, já que não é necessário mais do que reconhecer algumas frases mais simples e estereotipadas de cada língua para permitir aos alunos a aprendizagem e consolidação de determinada língua. Estas *Elizas* têm ainda mais potencial quando são associadas a um programa sintetizador de voz (*text-to-speech*). Isto permite que o aluno, para além de aprender o vocabulário, possa também aprender a sua dicção. Um projecto interessante com esta filosofia é o *Esteban* [Fer06], um projecto brasileiro para a aprendizagem da língua espanhola.

Outro dos projectos mais interessantes, a nível educacional, foi lançado pelo governo britânico, mais especificamente pelo NHS (*National Health Service*, o Serviço Nacional de Saúde Britânico) e tem o nome *Frank*. Consiste numa campanha anti-drogas, lançada em 2003, e é promovida através da televisão, rádio e Internet. A campanha, pela Internet, tem, além de uma página com perguntas e respostas, uma *Eliza*, construída para funcionar através cliente de mensagens instantâneas *Windows Live Messenger*, e que responde a perguntas dos utilizadores sobre drogas, tabaco, álcool e doenças relacionadas. No decorrer da campanha *Talk To Frank* foi conseguida a adesão da Eliza nos clientes de mensagens instantâneas de meio milhão de jovens britânicos.

Conversa 2.7: Exemplo de uma conversa com o *Talk to FRANK*.

Utilizador: Hello!
Talk to FRANK: Welcome back Utilizador!

The Talk to FRANK Bot is a fully automated Windows Live Agent designed to give young people information on drugs from content supplied by FRANK, the national drug information and advice service. Information provided to you by FRANK is not endorsed by Microsoft and does not reflect Microsoft's views on drug use. For more information, click here —> <http://l.rdir.us/ct?234869/0>

Here are some questions you can ask me:

1 Is cannabis harmless?

2 Is cannabis legal?

To find out more, type "?".

Utilizador: 1

Talk to FRANK: Even hardcore smokers can become anxious, panicky, suspicious or paranoid.

It affects your coordination, which is one of the reasons why drug driving is just as illegal as drink driving.

Some people think cannabis is harmless just because it's a plant – but it isn't harmless. Cannabis, like tobacco, has lots of chemical 'nasties', which can cause lung disease and possibly cancer with long-term or heavy use, especially as it is often mixed with tobacco and smoked without a filter. It can also make asthma worse, and cause wheezing in non-asthma sufferers.

Cannabis itself can affect many different systems in the body, including the heart: It increases the heart rate and can affect blood pressure.

If you've a history of mental health problems, then taking cannabis is not a good idea: It can cause paranoia in the short term, but in those with a pre-existing psychotic illness, such as schizophrenia, it can contribute to relapse.

If you use it and have a family background of mental illness, such as schizophrenia, you may be at increased risk of developing a psychotic illness.

It is reported that frequent use of it can cut a man's sperm count, reduce sperm motility, and can suppress ovulation in women and so may affect fertility.

If you're pregnant, smoking it frequently may have some association with the risk of the baby being born smaller than expected.

Regular, heavy use makes it difficult to learn and concentrate. Some people begin to feel tired all the time and can't seem to get motivated.

Some users may want to buy strong herbal it to get 'a bigger high' but unpleasant reactions can be more powerful when you use strong cannabis, and it is possible that using strong cannabis repeatedly could lead in time to more users experiencing harmful effects such as dependence or being more at risk of developing the mental health effects.

Utilizador: Is cannabis legal?

Talk to FRANK: Legal Status of Cannabis

-Cannabis is ilegal; it's a Class C drug.

-If you're caught with cannabis the police will always take action.

Type "more" to find out about possession and supplying and other facts.

Este tipo de *Elizas* tem como principal problema a sensação de inutilidade ou perda de tempo quando não é capaz de responder às perguntas apresentadas.

2.2.3 *Elizas* de atendimento ao consumidor

Os serviços de atendimento ao consumidor apresentam, geralmente, dois factores negativos para os consumidores: o tempo dispendido (por exemplo, aguardar que um funcionário de *call center* atenda a uma chamada) e a necessidade de se utilizar meios de comunicação específicos. Na tentativa de reduzir custos e aumentar a eficiência no apoio aos consumidores surgiram as *Elizas* de atendimento ao consumidor.

Exemplos internacionais de *Elizas* de atendimento ao consumidor são o *YODA* [Luc10], com o nome de uma das personagens dos filmes da saga *Star Wars*, desenvolvido pela empresa Lucas Arts, que tem como objectivo dar suporte técnico aos utilizadores dos seus jogos, ou o *HANK* [CC10], desenvolvido pela empresa *Coca-Cola* para responder às dúvidas dos utilizadores sobre a empresa e também sobre os seus produtos.

O mais famoso exemplo de uma *Eliza* de atendimento ao consumidor é o *Clippit*, também conhecido como *Clippy*, da Microsoft.

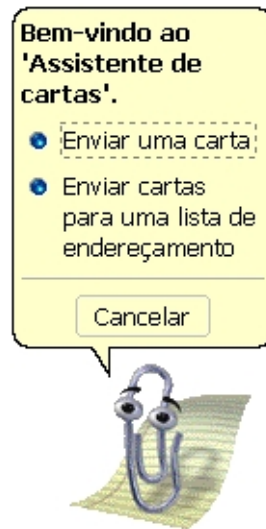


Figura 2.2: *Clippit*, o assistente do *Microsoft Office*.

Este famoso assistente, com a forma de um clip, funcionava como ajuda no *Microsoft Office*, e fornecia dicas ao utilizador sobre como realizar algumas operações nas ferramentas do *Office*. Por exemplo, caso se iniciasse um novo documento no *Microsoft Word* utilizando o modelo para cartas, o *Clippit* automaticamente perguntava ao utilizador se pretendia enviar uma carta ou enviar cartas para listas

de endereçamento. As dicas também surgiam por diversas vezes, explicando ao utilizador quais os atalhos que podia utilizar, assim como onde poderia encontrar determinada opção.

O *Clippit* surgiu no *Office 97*, e permaneceu em várias versões, tanto para *Windows*, como para *Macintosh*, tendo desaparecido nas versões *Office 2007* e *Microsoft Office 2008 for Mac*. A reacção dos utilizadores perante o assistente era extrema, pois tanto o adoravam, achando útil os seus conselhos e dicas, como o odiavam, dado o tempo que se perdia com o assistente.

Exemplos de *Elizas* portuguesas de atendimento ao consumidor são o *Buscas* [PAI10], assistente nas pesquisas nas *Páginas Amarelas*, e a *Anna* [IKE10], assistente na compra de móveis e artigos de decoração na companhia sueca *IKEA*.

O *Buscas* é uma *Eliza* que permite fazer pesquisas nas Páginas Amarelas, e utiliza o cliente de mensagens instantâneas *Live Messenger*. Tem uma sintaxe simples, que consiste num padrão “[O QUÊ] em [LOCAL]” para pesquisa de empresas ou serviços, e caso se pretenda repetir a pesquisa, mas com outra localidade, basta dizer “e no [LOCAL]”. Para pesquisar pessoas é necessário usar um prefixo de entre “Senhor”, “Senhora”, “Sr”, “Sra”, “Dr”, “Dr^a”, “Miss” e “Mister”.

O *Buscas* tem ainda como funcionalidade a indicação do local do contacto escolhido através da pesquisa num mapa, além de devolver todas as informações disponíveis no site das Páginas Amarelas⁸ ou das Páginas Brancas⁹.

Conversa 2.8: Exemplo de utilização do *Buscas*.

Utilizador: Olá

Buscas: Olá, eu sou o *Buscas*. Estou aqui para te ajudar a encontrar moradas e contactos de particulares (através das Páginas Brancas) e empresas (através das Páginas Amarelas).

Vou dar-te algumas dicas:

- Por ex. se quiseres encontrar farmácias em Lisboa pede-me "procura farmacias em lisboa"

- Se afinal queres farmácias em Coimbra basta dizeres "e em coimbra?"

- Caso estejas à procura do João nas Caldas da Rainha escreve "procuro o senhor joao nas caldas da rainha"

Se em qualquer altura tiveres alguma dúvida basta escrever "Ajuda".

O que procuras?

Utilizador: preciso do contacto do sr joao no porto

Buscas: Encontrei 50+ resultados para "joao"em "porto", aqui estão:

1. Adega João; R Revilão 351, Porto

2. João Aguiar; R Doutor Alves Veiga, Porto

(...)

> Escreve o número do resultado para mais informações.

> Escreve 'M', seguido de ENTER, para mais resultados.

⁸<http://www.pai.pt>

⁹<http://www.pbi.pt>

> Escreve 'E', seguido de ENTER para procurar por empresas.
(repete esta pesquisa).

> Escreve 'Ajuda' se tiveres dúvidas.

Finalmente, a *Anna* é uma *Eliza* disponível na página da *IKEA* e que adiciona alguma expressividade às suas funcionalidades. Quando se questiona a *Anna* acerca de um produto vendido naquela cadeia de lojas, ela automaticamente abre a página referente ao produto referido, conseguindo assim que o utilizador encontre mais facilmente os produtos disponíveis nas lojas *IKEA*. Além disso, a *Anna* consegue efectuar cálculos com somas, subtracções, multiplicações e divisões, além de responder a algumas perguntas pessoais que lhe sejam feitas.

Capítulo 3

Arquitectura da Etelvina

O processo de desenvolvimento da Etelvina foi modular, o que permitiu uma grande flexibilidade tanto na criação de novas funcionalidades, como na alteração de funcionalidades existentes.

De forma a permitir o crescimento do projecto, todos os desenvolvimentos feitos incidem na criação de novos módulos, módulos estes que têm o processamento de uma área específica, tendo como ponto de ligação o motor da Etelvina. É neste motor que são invocados os métodos de cada um dos módulos, que podem assumir que a mensagem é dirigida a eles e que estão em condições de dar uma resposta adequada a essa mensagem.

3.1 Estrutura de Implementação

A Etelvina foi desenvolvida com base num conjunto de pequenos módulos, que possibilitam a configuração da sua personagem/personalidade. Para que isso fosse possível, criou-se um bloco central denominado de motor, que é responsável pelo encaixe de novos módulos, sem ser necessário alterar o motor.

Sobre o motor funciona um conjunto de módulos com funcionalidades distintas, de forma a se poder moldar uma personagem para a Etelvina. Como se pode ver na figura 3.1, existe um módulo central (o motor da Etelvina) que fará a conexão dos módulos existentes com os clientes de mensagens instantâneas, que representam os utilizadores da Etelvina.

Um dos objectivos inicialmente traçado consiste na possibilidade da Etelvina ser flexível, sendo totalmente expansível e configurável. Assim, o resultado pretendido não consiste na obtenção de uma Eliza comum, em que o seu comportamento é sempre o mesmo, visto que está definido na base do código, mas sim um conjunto de Elizas, em que o comportamento deixa de se encontrar na base e passa a ser definido pelo utilizador.

Deste modo, decidiu-se pela implementação de uma estrutura composta por um motor, genérico, e um conjunto de módulos distintos, sendo apenas dependentes do

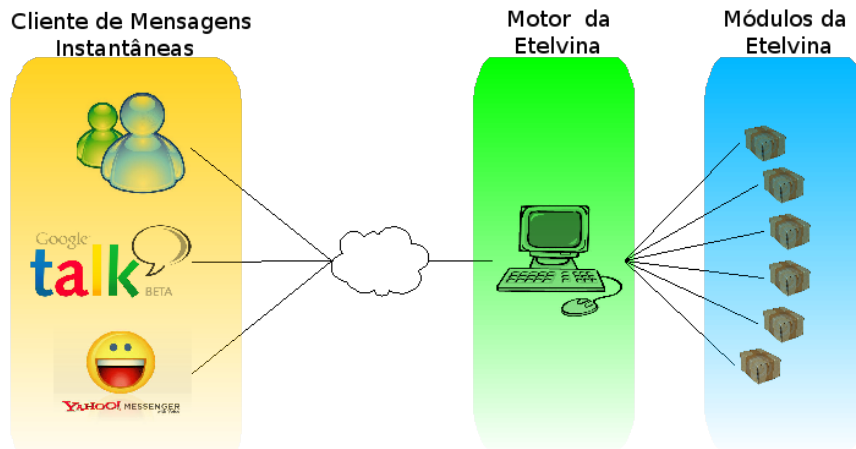


Figura 3.1: Representação da estrutura da Etelvina.

motor, módulos esses responsáveis por tarefas bem definidas.

Perante este cenário, fez-se uma análise do que era necessário existir para se poder obter a Eliza completamente configurável, de forma que os utilizadores consigam definir os seus comportamentos. Para isso, criaram-se dois tipos de configurações necessários para essa flexibilidade:

1. Configurações a nível do motor da Etelvina.

Alguns exemplos deste tipo de configurações são:

- a) Servidor/Cliente de *Instant Messaging* desejado;
- b) Definições da conexão a utilizar;
- c) Módulos a carregar e a ordem pelos quais serão invocados.

2. Configurações a nível de cada módulo.

Por exemplo, a definição de conjuntos de perguntas ou frases que serão reconhecidas por cada módulo, a forma de cálculo da resposta e sua composição numa frase de linguagem natural.

Tendo estes dois grupos de configurações consegue-se alterar o comportamento da Etelvina, visto que é possível alterar tanto definições relacionadas com a conexão a um servidor de mensagens instantâneas, como definir quais os módulos pretendidos, permitindo assim ao utilizador, através das configurações a nível de módulo, ter acesso às funções disponíveis em cada módulo, moldando o comportamento da Etelvina conforme o desejado.

3.1.1 Configurações a nível do motor da Etelvina

Tal como supracitado, o motor da Etelvina é a base sobre a qual foi construído todo o projecto.

De forma a ser o mais genérico possível, o motor está preparado para que se altere o protocolo de comunicação, onde, para efectuar tal alteração, apenas existe a necessidade de se mudar a API para se utilizar o novo protocolo desejado, e todo o restante funcionamento será exactamente o mesmo.

Este motor tem as seguintes funcionalidades:

- Controlar todo o fluxo de mensagens;
- Permitir ligar os módulos para responder às mensagens.

Controlar todo o fluxo de mensagens é um processo complexo e longo. O processo inicia-se com a autenticação no servidor de mensagens instantâneas a utilizar, permitindo que seja possível estabelecer a recepção e envio de mensagens. Depois é necessário saber quais os contactos da sua lista que estão disponíveis ou offline, assim como todas as alterações de estado dos contactos, de forma a não se enviar mensagens para utilizadores que estejam ausentes.

Após a autenticação e verificação de mensagens, a Etelevina está disponível para enviar e receber mensagens instantâneas. Quando se envia uma mensagem à Etelevina, automaticamente o motor guarda numa estrutura o remetente da mensagem em questão, assim como a hora a que se iniciou a conversa, a hora que recebeu a última mensagem, o número de mensagens recebidas naquela conversação, uma lista com as mensagens recebidas e o número de saudações recebidas.

utilizadorX@email.com	→	Utilizador	utilizadorY
utilizadorY@email.com		Hora início conversação	2009/10/01 10:30
		Hora da última mensagem	2009/10/01 11:03
...		Lista das últimas mensagens recebidas	("Olá", "Qual o teu nome?", "Quantos anos tens?",...)
		Número de mensagens recebidas	31
utilizadorZ@email.com		Número de saudações recebidas	1

Figura 3.2: Estrutura de informações armazenadas por utilizador.

Estas informações são armazenadas para que se consiga fazer algumas validações sobre as mensagens recebidas. Por exemplo, se a Etelevina receber uma mensagem

repetida nas últimas dez mensagens recebidas, responderá que já foi recebida uma mensagem igual, e por isso não devolverá novamente a resposta dada anteriormente. De frisar que a estrutura é armazenada por endereço associado ao cliente de mensagens instantâneas, o que permite que o controlo de cada conversa seja independente das outras conversas que a Etelevina tem presentes.

É também importante referir que quando o utilizador se desconecta do servidor de mensagens instantâneas, a Etelevina detecta essa acção e, conseqüentemente, elimina as informações referentes ao utilizador em questão. Assim, quando o utilizador se conectar novamente, a Etelevina vai criar uma nova estrutura, não tendo em conta a estrutura anterior.

Esta estrutura pode ser alterada pelo utilizador, de forma a adicionar mais informações que ache necessárias ou interessantes para a criação de uma personagem para a Etelevina.

Tendo o motor como funções controlar todo o fluxo de mensagens e interligar os módulos para responder às mensagens de entrada, foi necessário ter uma configuração que definisse todos os parâmetros variáveis de forma que o motor consiga ter o seu comportamento alterado, tanto a nível de controlo do fluxo de mensagens, assim como a nível de interligação dos vários módulos disponíveis.

Assim, a fase de desenvolvimento da configurabilidade do motor foi precedida por um levantamento de requisitos, levantamento esse feito após uma fase de análise. Tendo em conta as duas funções principais do motor, as configurações necessárias foram divididas em dois grupos principais: as configurações de ligação ao servidor de mensagens instantâneas e as configurações para definir quais os módulos que se pretende que sejam utilizados.

Após a fase de análise, em que se definiram os dois pontos chave para a criação da configuração, foi feito o levantamento de requisitos, para saber quais as informações que seriam necessárias ter como configuráveis.

Inicialmente, foi criada a configuração para a ligação ao servidor de mensagens instantâneas, tendo para isso definido-se como prioritário saber qual o protocolo que se pretende utilizar. Após se ter definido o protocolo, é necessário saber qual o servidor de mensagens instantâneas a utilizar, assim como a porta desse mesmo servidor. Por fim, é necessário configurar as informações de utilizador registado no servidor escolhido, tendo-se definido os seguintes parâmetros como configuráveis:

Protocolo - protocolo de mensagens instantâneas utilizado;

Username - nome de utilizador registado no servidor;

Password - palavra-passe do utilizador;

Status - mensagem de estado para mostrar aos outros utilizadores;

Displayname - nome a ser utilizado pelo contacto.

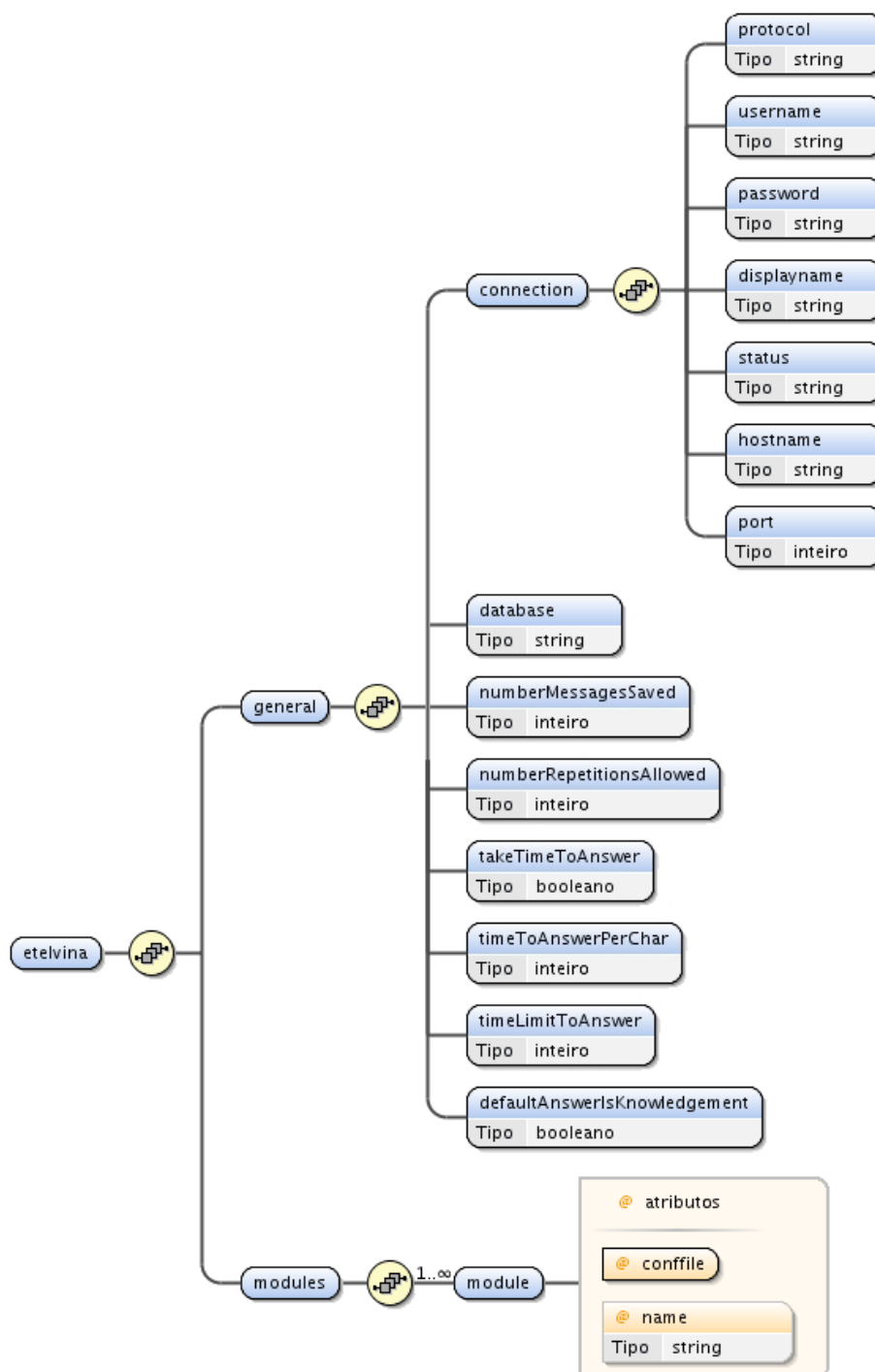


Figura 3.3: Schema de validação do XML de configuração do motor.

Com a possibilidade dos dados supracitados serem configuráveis, consegue-se assim efectuar ligações a servidores de mensagens instantâneas com vários utilizadores,

permitindo deste modo à Etelvina possuir, logo à partida, diferentes utilizadores e conseguir ser customizável por utilizador, sendo possível ter configurações de ligação a um servidor de mensagens instantâneas.

Para configurar quais os módulos que se pretende que sejam utilizados, teve-se em atenção que seria necessário, além de definir quais módulos pretendidos, a localização dos ficheiros de configuração, ficheiros esses que podem ser correspondentes a cada módulo ou simplesmente ficheiros de configuração de comportamentos que sejam independentes de módulos, configurações essas que serão explicadas mais pormenorizadamente no capítulo 3.1.2.

Para além destas configurações, foram também adicionadas duas configurações para definir comportamentos da Etelvina. Essas definições contêm se é pretendido pelo utilizador que a Etelvina utilize, em caso de não obter nenhuma resposta perante os módulos configurados, o módulo da Wikipédia para devolver uma resposta (ver capítulo 4 para mais informações), e também qual a base de dados a utilizar para o armazenamento de informações que provenham do módulo da Wikipédia.

Para definir todas estas informações, criou-se um esqueleto em XML, de forma a se possuir as informações numa estrutura de fácil compreensão para o utilizador, representado na figura 3.3.

De forma a poder utilizar as informações disponíveis no ficheiro de configuração, foi necessário transformar o ficheiro de configuração numa estrutura de dados de fácil acesso. Assim, criou-se uma estrutura principal, que é um array associativo, que possui duas chaves fixas correspondentes às etiquetas do ficheiro de configuração: *general* e *modules*.

Exemplo 3.1: DTD de validação do XML de configuração do motor.

```
<!ELEMENT etelvina (general,modules)>
<!ELEMENT general (connection,database,
    defaultAnswerIsKnowledgegement)>
<!ELEMENT modules (module)+>
<!ELEMENT connection (protocol,username,password,displayname,
    status,hostname,port)>
<!ELEMENT database (#PCDATA)>
<!ELEMENT numberMessagesSaved (#PCDATA)>
<!ELEMENT numberRepetitionsAllowed (#PCDATA)>
<!ELEMENT takeTimeToAnswer (#PCDATA)>
<!ELEMENT timeToAnswerPerChar (#PCDATA)>
<!ELEMENT timeLimitToAnswer (#PCDATA)>
<!ELEMENT defaultAnswerIsKnowledgegement (#PCDATA)>
<!ATTLIST defaultAnswerIsKnowledgegement xmlns CDATA #FIXED ''>
<!ELEMENT module EMPTY>
<!ATTLIST module conffile CDATA #REQUIRED name NMTOKEN #
    IMPLIED>
<!ELEMENT protocol (#PCDATA)>
```

```

<!ELEMENT username (#PCDATA)>
<!ELEMENT password (#PCDATA)>
<!ELEMENT displayname (#PCDATA)>
<!ELEMENT status (#PCDATA)>
<!ELEMENT hostname (#PCDATA)>
<!ELEMENT port (#PCDATA)>

```

O valor associado à chave *general* na estrutura principal corresponde a uma nova estrutura daquele tipo, de forma a conseguir armazenar as informações de qual a base de dados a utilizar (etiqueta *database*), se por defeito utiliza o módulo da Wikipédia (etiqueta *defaultAnswerIsKnowledge*), o número de mensagens recebidas que guarda em memória (etiqueta *numberMessagesSaved*), o número de repetições de frases permitido comparativamente às mensagens armazenadas, tomando um valor inferior a zero se não se pretende um limite (etiqueta *numberRepetitionsAllowed*), se a Etelevina deve demorar tempo a responder, simulando o tempo de escrita (etiqueta *takeTimeToAnswer*), qual o tempo, em segundos, por carácter que se pretende que a Etelevina demore para dar a resposta (etiqueta *timeToAnswerPerChar*, e apenas válido se a opção *takeTimeToAnswer* estiver activa), qual o tempo limite que uma resposta da Etelevina pode demorar (etiqueta *timeLimitToAnswer*, também ele dependente do valor da opção *takeTimeToAnswer*), e também as informações correspondentes a dados de conexão ao servidor de mensagens instantâneas, que por sua vez também é um array associativo, em que as chaves voltam a ser as etiquetas do ficheiro XML de configuração.

Exemplo 3.2: Exemplo de um ficheiro de configuração válido.

```

<etelvina>
  <general>
    <connection>
      <protocol>xmpp</protocol>
      <username>utilizadorX</username>
      <password>password</password>
      <displayname>Etelevina</displayname>
      <status>Supercalifragilisticexpialidous</status>
      <hostname>talk.google.com</hostname>
      <port>5222</port>
    </connection>
    <database>etelvinaDB.db</database>
    <numberMessagesSaved>15</numberMessagesSaved>
    <numberRepetitionsAllowed>2</numberRepetitionsAllowed>
    <takeTimeToAnswer>true</takeTimeToAnswer>
    <timeToAnswerPerChar>0.33</timeToAnswerPerChar>
    <timeLimitToAnswer>15</timeLimitToAnswer>
    <defaultAnswerIsKnowledge>
      true
    </defaultAnswerIsKnowledge>

```



```
</general>
<modules>
  <module name="Etelvina::Meteo"
          conffile="meteorologia.conf"/>
  <module conffile="personalidade.conf"/>
  <module name="Etelvina::MatExp" conffile="matExp.conf"/>
  <module name="Chess::Game" conffile="xadrez.conf"/>
</modules>
</etelvina>
```

Em relação ao valor correspondente à chave *modules* na estrutura principal, este corresponde a uma lista de módulos (assume-se como módulo um ficheiro de configuração associado a um módulo Perl ou isolado), módulos esses correspondentes a um array associativo, estrutura na qual se possui sempre o valor da localização do ficheiro de configuração desejado, assim como pode possuir o nome do módulo Perl pretendido para o ficheiro de configuração definido, de forma a poder utilizar as funções que aquele disponibiliza. Esta informação pode não existir, dado que não é obrigatório adicionar esta configuração no ficheiro de XML, pois a aplicação possibilita que não se utilize nenhum módulo base para se configurar um comportamento da Etelvina. No entanto, e de forma a se garantir que todas as referências para o módulo se encontram correctas, são gerados nomes únicos, criando assim uma chave para tais referências.

Ao definir a estrutura de dados para os módulos, recorreu-se às listas para se definir qual a ordem de processamento de módulos desejada. Essa ordem é igual à ordem das entradas dos módulos do ficheiro de configuração.

Para simplificar a compreensão de todo o processo de transformação de um ficheiro XML para a utilização das informações na aplicação, utilize-se como base o exemplo 3.2. Perante a configuração do exemplo, a estrutura de dados criada é a exemplificada na figura 3.4.

Como se pode verificar, a estrutura adoptada consegue armazenar todos os valores do ficheiro de configuração, com a vantagem de ser bastante simples aceder a qualquer um dos valores.

Perante os módulos configurados, correspondendo cada módulo a cada entrada do XML com a etiqueta *module*, tem-se, quando é recebida uma mensagem, o seguinte processamento:

Tendo em conta que a configuração tem o valor *true* para a etiqueta ***defaultAnswerIsKnowledge***, caso os vários módulos configurados não consigam devolver uma resposta após o seu processamento (ver figura 3.5), irá utilizar o módulo da Wikipédia para gerar uma resposta.

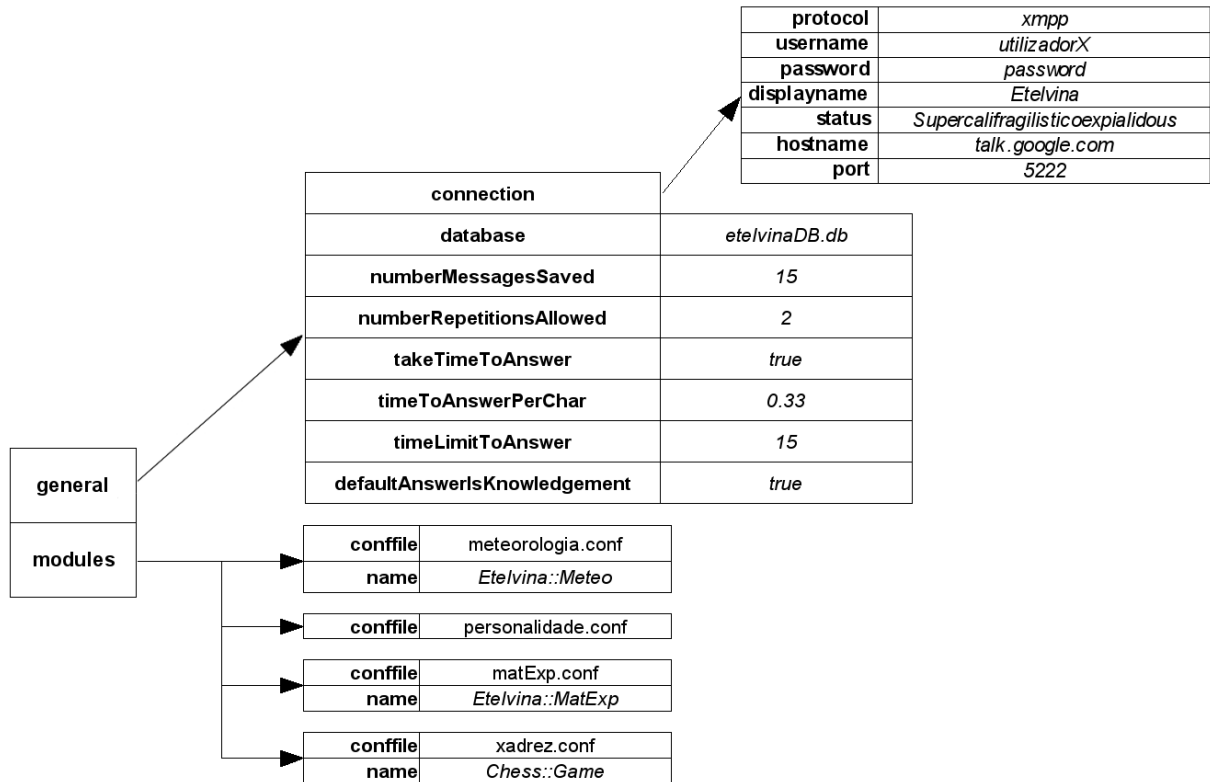


Figura 3.4: Esquema da estrutura de dados criada segundo o ficheiro de configuração representado no exemplo 3.2.

3.1.2 Configurações por módulo

De forma a se conseguir obter comportamentos da Etelvina completamente configuráveis, foi necessário criar um sistema de configurações por módulo, com a possibilidade de se definir perguntas a interpretar, respostas a utilizar e alterar estados a nível da aplicação. Para que qualquer utilizador possa configurar os comportamentos da Etelvina, o processo de configuração tem de ser relativamente simples.

Assim, foram definidas três características principais que dão origem à linguagem criada:

1. Capacidade de definição de novas mensagens que a Etelvina consiga interpretar;
2. Capacidade de definição de respostas para as mensagens de entrada previamente configuradas;
3. Capacidade de executar cálculos e alterar estados para possibilitar que determinadas acções (mensagens recebidas pela Etelvina) tenham efeitos além de simples reacções (responder a uma questão).

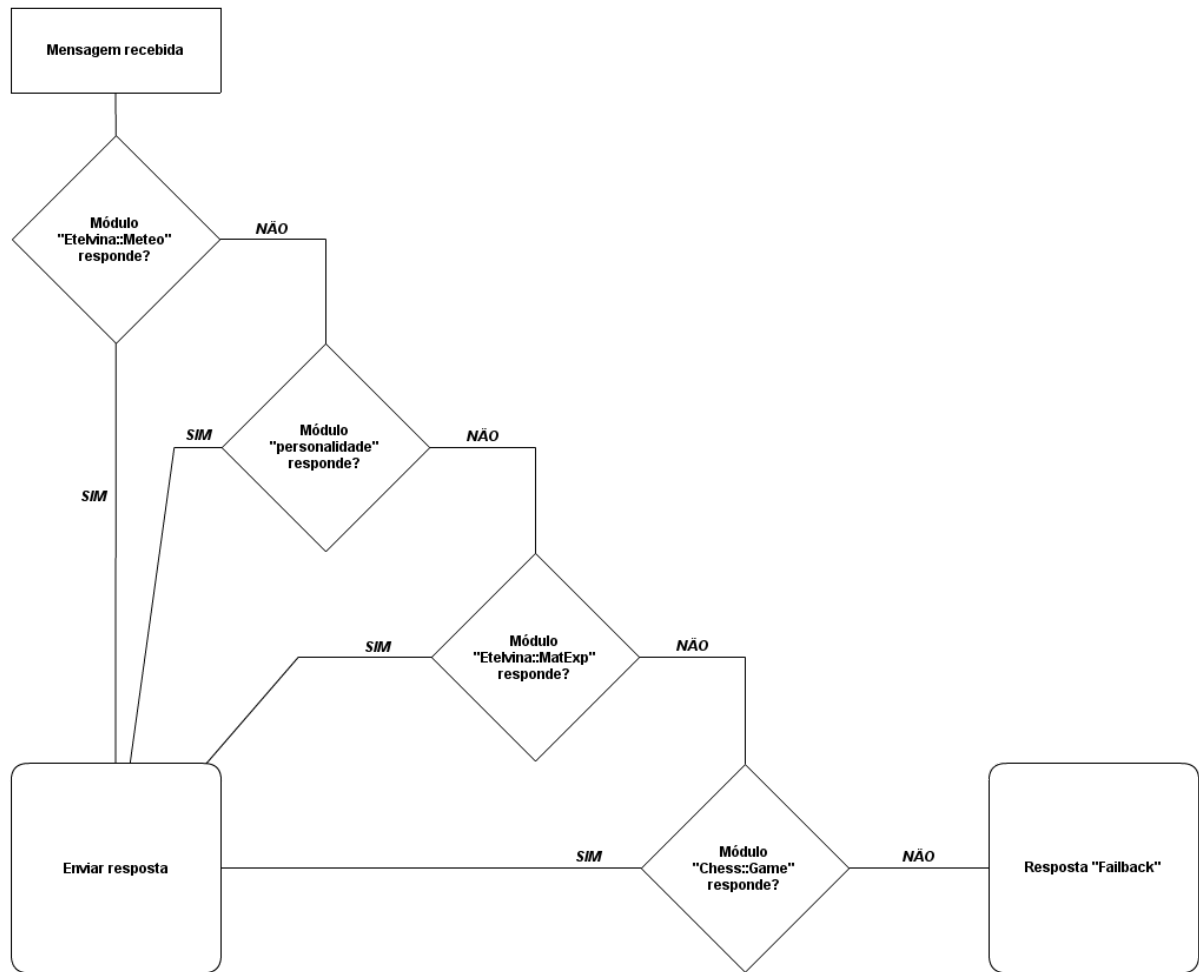


Figura 3.5: Ordem de processamento de módulos segundo a configuração.

Perante estes requisitos, criou-se uma linguagem que permite definir perguntas, de forma que a Etelvina reconheça novas mensagens, executar blocos de código ao reconhecer uma mensagem e devolver respostas. Assim, obteve-se um sistema de regras, todas elas definidas pelo utilizador.

Cada regra desta linguagem não é mais do que um sistema de reescrita. Antes de mais, foi definido como regra deste género de configurações o conjunto de mensagens de entrada possíveis (padrões de entrada), um bloco de instruções de cálculo e as respostas possíveis.

De forma a simplificar a configuração de um módulo, em vez de se poder ter apenas um padrão de match, pode-se ter vários, em que qualquer um que faça *match* despoleta o sistema, e a resposta é escolhida aleatoriamente dentro do conjunto de respostas possíveis. Assim, obtém-se a seguinte definição de regra:

$$regra = padrão^* \times código? \times resposta^*$$

Para se poder ter um módulo configurável, de forma a que se obtenha o comportamento desejado, na resposta a mensagens de entrada, criou-se um ficheiro que serve de esqueleto de suporte, de modo que seja possível configurar perguntas, ou seja, mensagens de entrada reconhecíveis pela Etelevina, os cálculos (blocos de código) necessários após o reconhecimento de uma determinada mensagem e as respostas possíveis.

Exemplo 3.3: Exemplo de uma entrada de configuração por módulo.

```
P: Qual a temperatura em <localidade,MWORD>?
P: Em <localidade,MWORD>, quantos graus estão?
C: $$valor = temperatura($$localidade);
R: Em <localidade> estão <valor>°.
R: Estão <valor>°C em <localidade>.
```

Como é possível verificar no exemplo 3.3, a linguagem adoptada possui uma sintaxe própria, sendo que mesmo sem conhecer essa linguagem, é perceptível que está configurada no exemplo uma regra, composta por duas questões, que são os padrões a reconhecer como mensagem de entrada, um bloco de código e duas respostas possíveis para a pergunta recebida.

Conversa 3.1: Exemplo de uma conversa com a Etelevina, perante a configuração definida no exemplo 3.3.

```
Utilizador: Qual a temperatura em Lisboa?
Etelevina: Em Lisboa estão 34°C.
Utilizador: Em Kuala Lumpur, quantos graus estão?
Etelevina: Estão 27°C em Kuala Lumpur.
Utilizador: Qual a temperatura em Freixo de Espada à Cinta?
Etelevina: Estão 32°C em Freixo de Espada à Cinta.
Utilizador: qual a TemPeratura Em Buenos Aires?
Etelevina: Estão 10°C em Buenos Aires.
```

Considerando os três pontos fundamentais para a configuração de um módulo supracitados, definiu-se uma linguagem, em que a estrutura básica de reconhecimento/cálculo/resposta a uma mensagem de entrada, encontra-se agrupada, numa visão simplista, pelos seguintes blocos e com a seguinte sintaxe:

1. **Bloco de perguntas**, com a sintaxe “P: *texto a reconhecer como entrada*”;
2. **Bloco de código a executar**, com a sintaxe “C: *bloco de código que deve ser executado*”;
3. **Bloco de respostas**, com a sintaxe “R: *resposta a devolver*”.

Tendo em conta a necessidade de se utilizar valores presentes na mensagem de entrada para poder devolver uma resposta correcta, criou-se um sistema de representação de variáveis no ficheiro de configurações para serem depois acessíveis quer na resposta, quer nos blocos de código definidos nesse mesmo ficheiro.

Assim, e tendo este género de variáveis como principal função poderem ser utilizadas de alguma forma para a criação de uma resposta, estas variáveis podem ser criadas quer a nível de configuração de mensagem de entrada, quer a nível de bloco de instruções de código.

Para a criação de uma variável no bloco de perguntas, adoptou-se a seguinte sintaxe:

<nome da variável , expressão regular>

Expressões Regulares — Resumo

A expressão regular tem a sintaxe de expressões regulares na linguagem Perl, em que os seguintes padrões individuais principais:

`\w` faz *match* a caracteres alfanuméricos, incluindo o carácter “_”. `\W` faz *match* a caracteres não-alfanuméricos;

`\d` faz *match* a caracteres numéricos. `\D` faz *match* a caracteres não-numéricos;

`\s` faz *match* a espaços em branco. `\S` faz *match* ao inverso;

`.` faz *match* a todos os caracteres até ao fim de linha.

Os caracteres `*`, `+` e `?` têm o seu significado habitual, ou seja, `*` significa zero ou mais ocorrências, `+` significa uma ou mais ocorrências e `?` significa zero ou uma ocorrência.

Por exemplo, para criar uma variável cujo seu nome será “*num*” e que o seu valor é um número inteiro positivo, a declaração desta variável no ficheiro de configuração seria representada pelo seguinte par nome de variável/expressão regular:

Exemplo 3.4: Exemplo de criação de uma variável com o nome *num* e com o valor a ser um número inteiro positivo.

<num, \d+>

De forma a simplificar a configuração para um utilizador básico, decidiu-se criar quatro expressões regulares, através das quais se conseguem criar variáveis sem se conhecer expressões regulares. As expressões criadas são:

1. **WORD**, que é simplesmente uma palavra (com a expressão regular `\s*\w+\s*`);

2. **MWORD**, que é representação de um conjunto de palavras (com a expressão regular $\backslash s * \backslash w + (\backslash s + \backslash w +)*$);
3. **NUM**, que representa um número (com a expressão regular $\backslash s * \backslash d + (. \backslash d +)? \backslash s *$);
4. **MATEXP**, que representa uma expressão matemática. Esta tem uma expressão regular mais complexa, que é composta pelas expressões regulares exemplificadas na gramática 3.1.

<i>p1</i> :	num	→	(-)? d+ (. d+)?
<i>p2</i> :	fun	→	'ln'
<i>p3</i> :			'log'
<i>p4</i> :			'sin'
<i>p5</i> :			'cos'
<i>p6</i> :			'tan'
<i>p7</i> :			'sqrt'
<i>p8</i> :			'exp'
<i>p9</i> :	operador	→	+
<i>p10</i> :			-
<i>p11</i> :			*
<i>p12</i> :			/
<i>p13</i> :	exp	→	(*num)* operador num)*
<i>p14</i> :	func	→	(* fun?)* exp
<i>p15</i> :	math	→	func+

Gramática 3.1: Gramática de expressões matemáticas.

Tendo o nome da variável definida e também a expressão regular correspondente, é necessário fazer um tratamento para que essa variável esteja disponível nos blocos seguintes, na configuração de uma resposta para a mensagem de entrada em questão.

Esse tratamento consiste em, utilizando um array associativo, e ao fazer *match* a uma mensagem com a expressão criada (texto e expressões regulares que se pretende que os seus valores sejam utilizados em variáveis), inserir nessa estrutura o valor da expressão regular criada com a chave a ser igual ao nome atribuído à variável. Assim, e utilizando o exemplo 3.4 da variável “num”, entrando uma mensagem com o texto “12345”, e tendo em conta que faria *match* com a expressão regular definida ($\backslash d+$), iria ser acrescentado um valor na estrutura com a chave “num” e o valor correspondente igual a “12345”.

Assim, passa a ser possível utilizar os valores armazenados nos blocos seguintes de uma regra. Para utilizá-los nesses blocos é necessário utilizar também uma linguagem específica.

Conversa 3.2: Exemplo de uma conversa com a Etelevina utilizando as expressões matemáticas definidas na gramática 3.1.

```
Utilizador: 1+1?  
Etelevina: 2.  
Utilizador: exp 2 * ((3*5)/23)  
Etelevina: 3.68528486315213.  
Utilizador: log 10  
Etelevina: 2.30258509299405.  
Utilizador: log(10)  
Etelevina: 2.30258509299405.  
Utilizador: sqrt(log sin(3.2)*cos(23)/tan(12)+5)  
Etelevina: 1.26475553344411.  
Utilizador: exp 2  
Etelevina: 7.38905609893065.
```

Um bloco de código de uma regra não é mais do que a escrita de código na linguagem Perl, com algumas alterações, de forma a ser possível utilizar as variáveis criadas pelo sistema supracitado, assim como a utilização da estrutura referida na secção 3.1, para poder adicionar todo o tipo de controlos desejados.

As regras para a criação destes blocos de código são simples, e consistem nos seguintes pontos:

- Dado que o array associativo, onde se pode adicionar variáveis através do bloco de reconhecimento de mensagens de entrada, está armazenado na variável com o nome “hash”, não se deve apagar nenhum dado dessa estrutura, de forma a se garantir o correcto funcionamento da Etelevina. No entanto, este array associativo deve ser utilizado para alteração de valores de variáveis e criação destas, pois é a forma de utilizar variáveis nas respostas que estão configuradas;
- Tendo em conta que em Perl as variáveis escalares simples são declaradas com o carácter “\$” antes do nome da variável, para se poder diferenciar as variáveis criadas no bloco de mensagens de entrada do ficheiro de configuração, utiliza-se o nome da variável antecedido por “\$\$”. Assim, ao se utilizar o nome da variável precedido por “\$\$”, o que se está a fazer na realidade é utilizar o valor do array associativo criado com chave igual ao nome da variável;
- Para utilizar a estrutura por utilizador exemplificada na figura 3.2, basta referir a variável “\$estrutura”. Assim, e tal como no caso da variável “\$hash”, não é recomendável apagar nenhum elemento desta estrutura.

Desta forma, transformado em código Perl, e tendo em conta que a variável do array associativo tem o nome “hash”, tem-se as transformações representadas na tabela 3.1.

Código no ficheiro de configuração	Correspondência em código Perl
<pre>\$oMeuNome = \$\$nome; \$\$nome = "Etelvina"; \$nome = "Etelvina";</pre>	<pre>\$oMeuNome = \$hash->{nome}; \$hash->{nome} = "Etelvina"; \$nome = "Etelvina";</pre>

Tabela 3.1: Exemplos de algumas transformações de código do ficheiro de configuração para código Perl.

Sabendo que a localização deste ficheiro de configuração é definida no ficheiro de configuração a nível de motor, tal como é visível no exemplo 3.2, através da localização do nome do ficheiro (obrigatório) e o módulo Perl correspondente (opcional), este último é automaticamente importado, estando assim disponíveis, sem ser necessário importar manualmente o módulo desejado, todas as funções que o módulo dispõe.

A outra opção, ao invés de se adicionar a referência ao módulo desejado no ficheiro de configuração, seria obrigar o utilizador a adicionar sempre o módulo manualmente no bloco de código. No entanto, seria obrigatório ao utilizador ter um conhecimento mínimo da linguagem de programação Perl de forma a poder saber importar um módulo. Assim, utilizando apenas a notação supracitada para configurar cálculo na Etelvina, torna-se mais acessível conseguir utilizar funções de um determinado módulo. Apesar de incluir o módulo automaticamente, nada na Etelvina impede a importação de módulos, sendo possível adicionar os módulos desejados, de forma a se poder ter uma configuração de blocos de código com mais potencialidades.

Após a explicação de como se processa a criação de variáveis no bloco de configuração de mensagens de entrada e correspondente utilização destas no bloco de código, falta explicar como é feito o acesso às variáveis na configuração de blocos de resposta.

Para a utilização do valor de uma variável numa resposta, basta utilizar a seguinte sintaxe:

<nome da variável>

Quando a Etelvina está a “sortear” a resposta dada, o comportamento que tem é de substituir todas as ocorrências de *<variável>* pelo valor que a variável tem no array associativo criado. Caso a variável não exista, a substituição é feita por uma string vazia, pelo que apenas devem ser utilizados nomes de variáveis que realmente existam na estrutura referida.

Após a criação e utilização de variáveis no sistema, resta o restante texto que é necessário reconhecer. Para configurar um texto de forma a que a Etelvina reconheça, basta escrever normalmente a frase que se pretende que seja aceite como mensagem de entrada válida para um determinado módulo. O sistema, após a leitura do ficheiro de configuração, cria automaticamente uma expressão regular para

Frase a reconhecer	Expressão regular gerada
Hallo	<code>\s*hallo\s*</code>
Bons dias!	<code>\s*bons\s+dias\s*!\s* \s*</code>
What's your name?	<code>\s*what\s*' \s*s\s+ your \s+name\s*/?\s*/\s*</code>
Wie alt bist du?	<code>\s*wie\s+alt\s+ bist\s+du\s*/?\s*/\s*</code>
Isto é uma expressão regular, que apenas serve para testar a geração de novas expressões regulares, em que se ignoram os acentos	<code>\s*isto \s+ [ée]\s+uma \s+ express[ãa]o \s+ regular\s*,\s* \s+ que \s+ apenas \s+ serve \s+ para \s+ testar \s+ a \s+ gera[çc][ãa]o \s+ de \s+ novas \s+ express[õo]es \s+ regulares\s*,\s* \s+ \s+ em \s+ que \s+ ignoro \s+ os \s+ acentos \s+ tipo \s+ nos \s+ [âa]s \s+ e \s+ as \s+ tremas \s+ nos \s+ [üu]s\s*,\s* \s+ assim \s+ como \s+ os \s+ [çc]\s*</code>

Tabela 3.2: Tabela que exemplifica as expressões regulares geradas para algumas frases de entrada.

poder fazer *match*. No entanto, ignora a capitalização da mensagem de entrada, assim como múltiplos espaços. Uma vez que, em conversações através de mensagens instantâneas, a falta de acentos é relativamente comum, decidiu-se, de forma a ser menos rígido o sistema de *match* de mensagens de entrada, ignorar, além da capitalização, a acentuação de caracteres e também o carácter “ç”. Assim, uma mensagem de entrada com palavras sem qualquer acentuação e que no ficheiro de configuração sejam acentuadas são aceites para aquela expressão regular.

Suponha-se que a Etelvina tem configurado um módulo de temperaturas, em que o ficheiro de configuração do módulo é o representado no exemplo 3.3. Ao receber uma mensagem com o texto “em Lisboa, quantos graus estao?”, o comportamento da Etelvina é o seguinte:

1. Testa se a mensagem recebida faz *match* com alguma das seguintes expressões regulares, criadas a partir do ficheiro de configuração do exemplo 3.3:

`Qual\s+a\s+temperatura\s+em\s+(?<localidade>\s*|w+(|s+|w+)*|?)`
ou
`Em\s+(?<localidade>\s*|w+(|s+|w+)*),\s+quantos\s+graus\s+est[ãa]o|?`

É necessário saber que a capitalização é ignorada, além de que os blocos do tipo `(?<nomeVariavel> expressãoRegular)` representam a gravação na variável com o nome “nomeVariavel” e valor correspondente ao valor que a “expressãoRegular” fez *match*, que, no caso mensagem de entrada do exemplo, é “Lisboa”;

2. Caso faça *match* a alguma, executa o bloco de código definido no ficheiro de configuração;

```
$$valor = temperatura('Lisboa');
```

3. Por fim, utilizando os valores das variáveis “localidade” e “valor”, a Etelvina vai sortear qual das respostas configuradas irá responder, fazendo por último a substituição das variáveis pelos respectivos valores, ficando assim com uma resposta do género “Em Lisboa estão 18°C”.

Perante as possibilidades de configuração de um módulo, e considerando como acção a recepção de uma mensagem de um utilizador e reacção a devolução de uma resposta à mensagem inicial, os tipos de regras que se podem configurar resumem-se em três tipos, conforme os passos executados para responder à mensagem recebida, como se pode verificar nas secções seguintes.

Regras de acção-reacção

As regras de acção-reacção são simples e resumem-se a regras em que a configuração tem configurado como mensagens de entrada frases completas, em que eventualmente possuem expressões regulares para armazenar algum valor. Além disso, não possuem bloco de código e as respostas consistem em frases também elas completas ou com utilização de variáveis criadas na mensagem de entrada.

Exemplo 3.5: Exemplo de regras do tipo acção-reacção.

P: Olá	P: O meu nome é <nome ,MWORD>.
P: Oi	R: Olá <nome>!
R: Olá	
R: Olá, tudo bem?	

Estas mensagens têm a vantagem de serem muito simples de configurar, no entanto resumem-se a receber uma mensagem que reconhecem e devolver directamente uma resposta.

Conversa 3.3: Exemplo de uma conversa através de regras de acção-reacção.

```
Utilizador: ola
Etelvina: Olá
Utilizador: Oi
Etelvina: Olá, tudo bem?
Utilizador: O meu nome é Utilizador.
Etelvina: Olá Utilizador
```

Dependendo da quantidade de regras definidas com este tipo, podem-se obter resultados diversificados: se a quantidade de frases reconhecidas for baixa, a Etelvina

fica com uma base de conhecimento muito pequena, que se vai implicar a repetição constante de respostas, de forma muito “programada”; no entanto, se o número de regras for grande, a capacidade de interpretação de mensagens de entrada será grande, assim como as respostas serão mais variadas. Para se poder ter muitas mensagens de entrada reconhecidas pela Etelevina implica que o o utilizador teve de acrescentar muitas combinações, o que torna custoso a criação de muitas regras.

Regras de acção-computação-reacção

As regras de acção-computação-reacção, comparativamente com as regras de acção-reacção, exigem mais conhecimento por parte do utilizador, dada a componente de programação existente. Com isso, o utilizador que configure este tipo de regras necessita de conhecimentos da linguagem de programação Perl, assim como as particularidades de utilização de variáveis e estruturas existentes na Etelevina (ver secção 3.1.2), conseguindo assim criar blocos de código mais ou menos complexos, dependendo dos conhecimentos do utilizador. No entanto, este tipo de regras permite uma maior complexidade na devolução de respostas, assim como o acesso a serviços remotos, de forma a se conseguir ter respostas mais completas e reais, casos que não são possíveis com regras do tipo acção-reacção.

Tenha-se em conta o seguinte ficheiro de configuração, definido no exemplo 3.3.

```
P: Qual a temperatura em <localidade,MWORD>?
P: Em <localidade,MWORD>, quantos graus estão?
C: $$valor = temperatura($$localidade);
R: Em <localidade> estão <valor>°C.
R: Estão <valor>°C em <localidade>.
```

Assumindo que a função *temperatura* devolve o valor da temperatura, em graus Celsius, na localidade armazenada na variável “localidade”, consegue-se assim devolver uma resposta a uma pergunta que é temporalmente variável. Para isso, acede-se a um serviço online e constrói-se uma resposta com um valor válido para a questão do utilizador.

Outro dos casos em que é útil este género de regras é para actualização de estados da Etelevina. Relembrando a estrutura de dados que cada utilizador possui na Etelevina, descrita na secção 3.1.1, existe por defeito um campo com o número de saudações recebidas. Este campo foi adicionado na estrutura para que seja mais simples fazer um controlo do número de saudações que a Etelevina recebe de um determinado utilizador. No entanto, este dado não é preenchido pelo motor da Etelevina, pois este não tem a capacidade de reconhecer se uma mensagem é uma saudação ou não. Assim, para cada regra que tenha como mensagem de entrada uma saudação, se for pretendido que exista controlo do número de saudações enviadas pelo utilizador, deve-se actualizar o estado da Etelevina para o utilizador que escreve a saudação.

Exemplo 3.6: Exemplo da actualização do estado da Etelvina para um utilizador.

```
P: Olá  
C: $estrutura->{numSaudacoes}++;  
R: Olá!
```

Como é possível verificar no exemplo 3.6, é utilizada uma regra de acção-computação-reacção, onde o único objectivo é actualizar o estado da Etelvina.

Assim, com regras de acção-computação-reacção, consegue-se definir respostas que têm uma validade temporal muito curta, mas que, ao serem consultados os dados em tempo real, se mantêm válidas. Além disso, permitem a actualização de estados, de forma a se conseguir que a Etelvina tenha em consideração todos esses estados ao ter uma conversa.

Regras de acção-computação-reacção computada

Por fim, e não menos importante, têm-se as regras de acção-computação-reacção computada. Este tipo de configuração de regras permite ao utilizador criar respostas que provêm da computação, e que não necessitam de mais nenhum texto. Para isso, toda a computação tem de ser bem definida para que a resposta tenha lógica.

De forma a demonstrar como pode ser utilizado este género de regras, suponha-se o seguinte caso: ao configurar a Etelvina, pretende-se que, ao receber uma saudação, a Etelvina tivesse um comportamento variável, que consiste nos seguintes pontos:

- Caso o número de saudações seja superior a três, a Etelvina deve responder de forma irónica;
- Caso a diferença de tempo entre a primeira mensagem recebida de um utilizador e uma mensagem com uma saudação reconhecida pela Etelvina, enviada pelo mesmo utilizador, for superior a dois minutos, a Etelvina deve responder que já se encontra nessa conversa há algum tempo;
- Se o número de mensagens recebidas antes da saudação for superior a cinco, a Etelvina tem de alertar o utilizador que a conversa já tem alguma extensão;
- Se nenhuma das condições anteriores se verificar, a Etelvina responde com uma saudação entre as seguintes: “Olá”, “Boas”, “Bons olhos te vejam” e “Oi”, mensagens estas que são as que deverá reconhecer como saudações em mensagens de entrada.

Assim, com um sistema que apenas suporte regras com mensagens de entrada e consequente resposta não seria possível esse tipo de comportamento, sendo possível com o sistema adoptado. Tendo em conta que na estrutura por utilizador existente no sistema da Etelvina, tal como o demonstrado na figura 3.2, possui a hora de início da conversa, a hora da última mensagem recebida, o número de mensagens

recebidas e também o número de saudações recebidas, pode-se utilizar esses valores para criar testes no bloco de código de uma regra, de forma a ter o comportamento desejado. A forma de definir estas regras é a definida no exemplo 3.7.

Exemplo 3.7: Exemplo de configuração de um módulo com respostas variáveis.

```
P: Olá
P: Oi
P: Boas
P: Bons olhos te vejam
C: $estrutura->{numSaudacoes}++;
   my @mensagens = ( "Olá","Oi","Boas!","Bons olhos te vejam");
   my $first_msg_time = $estrutura->{initHour};
   my $last_msg_time = $estrutura->{lastMsg};
   my $diferencaTempo =
       diffTimeInSeconds($last_msg_time, $first_msg_time);
   if ($estrutura->{numSaudacoes} > 3) {
       $$valor = "Lol!";
   } else if ($diferencaTempo > 120 ) {
       $$valor = "Já estamos a conversar há algum tempo!";
   } else if ($estrutura->{msgCount} > 5 ) {
       $$valor = "Acho que já recebi várias mensagens tuas!";
   } else {
       my $range = scalar @mensagens;
       my $random_number = int(rand($range));
       $$valor = $mensagens[$random_number];
   }
R: <valor>
```

No exemplo 3.7 consegue-se cumprir todas as regras supracitadas. Assim, e utilizando as capacidades do sistema de configurações de módulos, consegue-se ter respostas variáveis segundo determinados dados/valores. Isto é possível armazenando a resposta desejada numa variável, em que o texto de resposta será apenas essa variável.

Conversa 3.4: Exemplo de uma conversa utilizando as regras definidas, com indicações temporais, no exemplo 3.7.

```
[20:06:36] Utilizador: Tudo bem?
[20:06:40] Etefvina: Sim, e contigo?
[20:06:48] Utilizador: Comigo também está tudo bem!
[20:06:53] Etefvina: Fico feliz por isso!
[20:07:11] Utilizador: O resultado de 3*5+12*4 é...
[20:07:11] Etefvina: 63.
[20:07:36] Utilizador: E 74*20+40*12+12+32-100?
[20:07:37] Etefvina: 1904.
[20:07:56] Utilizador: És muito inteligente!
[20:08:01] Etefvina: Obrigado! Um elogio sabe sempre bem!
```

[20:08:15] Utilizador: Olá!
[20:08:20] Etelevina: Acho que já recebi várias mensagens tuas!
[20:08:25] Utilizador: Oi!
[20:08:30] Etelevina: Acho que já recebi várias mensagens tuas!
[20:08:38] Utilizador: Oi!
[20:08:43] Etelevina: Já estamos a conversar há algum tempo!
[20:08:52] Utilizador: Olá!
[20:08:53] Etelevina: Lol!

Potencialidades do sistema de configuração adoptado

O sistema adoptado apresenta como principal característica a configurabilidade. Isto porque todos os comportamentos da Etelevina podem ser configurados a nível de cada módulo.

Ao se permitir que as mensagens de entrada que o sistema reconhece sejam configuráveis, a quantidade de mensagens que o sistema reconhece é facilmente expansível. Tem-se também a simplicidade de configurar uma nova regra, pois qualquer utilizador que conheça a sintaxe de escrita de regras consegue criar as suas regras.

A possibilidade de ter respostas dinâmicas, através de código, também se apresenta como uma funcionalidade com bastante potencial, dada a versatilidade de respostas que permite que a Etelevina devolva.

3.1.3 Carregamento e utilização de módulos

Uma das funções do motor da Etelevina consiste na interligação dos módulos para ter a capacidade de responder a mensagens, tal como fora supracitado. Para interligar os vários módulos, é necessário ter em atenção os seguintes pontos:

- a) O motor tem de saber utilizar todos os módulos que estão configurados;
- b) Tem de ser respeitada a ordem pela qual os módulos estão configurados no ficheiro de configuração do motor.

O carregamento de um módulo implica transformar todas as regras definidas para um módulo em código fonte, de forma a ser posteriormente utilizado, para se ter uma Etelevina que tenha os comportamentos de resposta desejados.

Utilizando a estrutura representada na figura 3.4, tem-se acesso aos módulos que estão configurados a nível de motor e que, portanto, serão utilizados pelo motor da Etelevina para conseguir criar a personagem de mensagens instantâneas.

Assim, quando se pretende carregar o módulo, inicialmente verifica-se qual o ficheiro de configuração onde estão definidas as regras criadas pelo utilizador. Após a abertura desse ficheiro, inicia-se a transformação das regras definidas.

Tendo em conta a sintaxe adoptada para a configuração individual de módulos (ver secção 3.1.2), a transformação ocorre utilizando um sistema de reescrita de regras. Assim, assumindo que todo o ficheiro de configuração se encontra correctamente escrito, existem padrões para cada regra, de forma a poder armazenar essa mesma regra e posteriormente utilizá-la quando for necessário.

Não é feita nenhuma validação a nível de sintaxe do ficheiro de configuração de cada módulo. Caso o ficheiro não cumpra as regras definidas, a aplicação não devolve erros. No entanto, não será possível garantir que as regras erradamente definidas tenham os comportamentos desejados. Assim, e para uma fase posterior, existe necessidade de criar um validador de ficheiros de configuração (ver secção 5.1).

A reescrita de regras é feita através do módulo “`Text::RewriteRules`” [Alm10], disponível para a linguagem Perl. Este módulo permite reescrever regras escrevendo a regra em duas partes: a parte que contém a expressão regular que se pretende que faça *match* no texto de entrada e a sequência de caracteres que se pretende utilizar para substituir o conteúdo que fez *match* com a primeira parte da regra.

A decisão de utilizar um sistema de reescrita de regras surgiu de forma natural, pois de alguma forma seria necessário criar expressões regulares para conseguir extrair as regras definidas no ficheiro de configuração. No entanto, fazer todo o processamento sem o auxílio de um sistema de reescrita de regras tal como o utilizado seria um processo mais custoso, em que os tempos de implementação seriam maiores, assim como a complexidade das transformações seria muito maior, tal como a dificuldade de leitura dessas regras.

Antes de definir as regras para transformar o ficheiro de configuração, foi necessário criar uma estrutura para armazenar todas as regras definidas. Assim, e tendo em conta o ficheiro de configuração, era necessário armazenar, para cada regra, os seguintes blocos:

1. Bloco de mensagens de entrada que a Etelvina deverá reconhecer;
2. Bloco de código que é executado quando alguma mensagem do bloco anterior é reconhecida;
3. Bloco de respostas, de onde a Etelvina sorteia a resposta a devolver à mensagem de entrada.

Perante a necessidade de armazenar cada um dos blocos de configuração de uma regra, definiu-se o seguinte:

- O bloco de mensagens de entrada é transformado numa lista, em que cada entrada corresponde a uma mensagem de entrada configurada;
- O bloco de código é armazenado numa variável de texto;
- O bloco de mensagens de resposta é transformado, tal como o de mensagens de entrada, numa lista, onde cada possibilidade de resposta é uma entrada.

Assumindo que uma regra é transformada em duas listas e uma variável textual, e que, de forma a se poder aceder mais facilmente a cada uma das partes constituintes de uma regra, estão armazenadas num array associativo, pode-se definir que um ficheiro de configuração a nível de módulo é simplesmente uma lista de regras.

Após a definição da estrutura de dados a suportar, resta saber quais as regras de reescrita e como preencher essa estrutura, de forma a se conseguir obter o comportamento desejado.

Para se armazenar os dados do ficheiro de configuração na estrutura desejada, decidiu-se que se iria preencher a informação textualmente em código-fonte Perl, e depois transformar esse mesmo código fonte em código executável, de forma a ser possível utilizar as estruturas definidas.

O processo de transformação do ficheiro de regras na estrutura de dados definida inicia-se com a abertura do ficheiro de configurações em modo texto. O primeiro tratamento consiste na remoção de todos os caracteres “ \n” em todo o texto, caracteres esses que correspondem a mudanças de linha, desde que é iniciado o bloco de código, marcado pela escrita do texto “C:” no início de uma linha. Assim, todo o código fica numa única linha de texto, visto que é uma necessidade que será explicada na secção das regras de reescrita.

Após essa remoção de caracteres de nova linha, será feita mais uma transformação. Essa transformação consiste na alteração de todas as ocorrências de regras que não possuam bloco de código, de forma a acrescentar um bloco de código que não executa qualquer acção. Esta alteração prende-se com a necessidade de se ter padrões fixos, de forma a se poder generalizar a transformação de regras na estrutura definida, assim como a possibilidade de se utilizar sempre da mesma forma essa mesma estrutura.

Estando o texto já todo formatado e regularizado, passa a ser possível definir regras de reescrita para a criação da estrutura de suporte a um ficheiro de configuração a nível de módulo, tal como supracitado. Para compreender melhor as regras de reescrita definidas, é necessário conhecer alguns conceitos de Perl.

Conceitos base para criação de variáveis e estruturas de dados em Perl

Na linguagem Perl, existem três tipos de variáveis que se podem criar:

- a. Variáveis escalares simples, que são definidas pelo nome da variável antecedido do símbolo \$;
- b. Variáveis que representam listas, que são definidas pelo nome da variável antecedido do símbolo @;

- c. Variáveis que representam arrays associativos, que são definidas pelo nome da variável antecedido do símbolo %.

Estrutura	Exemplos
<i>Lista:</i>	<code>@var = (valor1,valor2,valor3);</code>
<i>Referência para lista:</i>	<code>\$var = [valor1,valor2,valor3];</code>
<i>Array associativo:</i>	<code>%var = (chave1, valor1, chave2, valor2);</code>
<i>(Ou):</i>	<code>%var =(chave1 => valor1, chave2 => valor2)</code>
<i>Referência para array associativo:</i>	<code>\$var = {chave1, valor1, chave2, valor2}</code>

Uma vez que o texto já se encontra com a formatação requirida, para se iniciar o processo de reescrita das regras definidas falta apenas preparar a estrutura definida para armazenamento das regras. Tal como anteriormente explicado, a estrutura utilizada para o armazenamento das regras do ficheiro de configuração do módulo é composta por uma lista de arrays associativos, em que cada entrada dessa lista corresponde à estrutura composta por três tipos de valor: *lista de mensagens de entrada*, que tem como chave o texto “perguntas”, *o bloco de código*, com a chave “codigo” e por fim a *lista de perguntas*, com a chave “respostas”.

Assim é necessário, antes de se iniciar a reescrita de regras, declarar uma referência para uma lista. Por uma questão de simplicidade, criou-se uma variável que representa a referência para a lista que se pretende criar. Nessa lista, criou-se também uma entrada, correspondente a uma regra definida no ficheiro de configuração, e nesse array associativo coloca-se a chave “perguntas”, pois é a primeira lista que será preenchida, uma vez que as regras de reescrita são aplicadas sequencialmente, à medida que os padrões definidos para a transformação são encontrados, e portanto, dada a obrigatoriedade de se iniciar sempre a definição de uma regra no ficheiro de configuração com o bloco de mensagens de entrada a reconhecer (ver secção 3.1.2 para ver a sintaxe adoptada para a configuração de módulos). Para a chave “perguntas” é também atribuída inicialmente, como valor, uma lista vazia, de forma a ser preenchida na reescrita de texto.

Assim, assumindo que o carácter “.” representa o operador de concatenação de texto, quando se inicia o processo de reescrita de regras, possui-se o seguinte texto:

```
[ perguntas => [ ' . texto a ser tratado . ' ] ];
```

Este texto será o utilizado para definir a estrutura e seus dados, sendo escrita como texto para posteriormente, após a finalização do processo de reescrita de regras, ser transformado em estruturas carregadas em memória, de forma a poder ser utilizado na aplicação.

Tendo já criada a estrutura base que vai armazenar as regras definidas, pode-se iniciar o processo de reescrita. O processo de reescrita devolve, para cada regra que seja transformada, um resultado textual, que irá substituir o *texto a ser tratado* visível na string que se possui aquando do início do processo.

A ordem pela qual as regras de reescrita surgem têm influência, pois as regras são percorridas sequencialmente e após se conseguir aplicar uma regra, o bloco de texto é transformado por essa mesma regra e o processamento segue com o bloco de texto seguinte.

A primeira regra de reescrita definida é aplicada quando é encontrado um texto iniciado com os caracteres “P:”, seguido de texto livre até encontrar o fim de linha. Este texto é uma entrada do bloco de mensagens de entrada que deverá ser reconhecido pela Etelvina. Esta regra consiste na transformação de cada pergunta numa expressão regular, em que é ignorada a capitalização do texto, e em que também são tratados de forma especial os caracteres acentuados, de forma a serem aceites as palavras com ou sem acentuação, tal como é referido na secção 3.1.2. Além disso, também são feitas substituições de todos os blocos que definem uma variável no bloco de mensagens de entrada (ver secção 3.1.2), de forma a se transformar a sintaxe de criação de uma variável no ficheiro de configuração de regras para a sintaxe Perl.

Tendo em conta que podem existir várias linhas de uma mesma regra de configuração que comecem pelo carácter “P”, ao fim de cada *match* de uma regra deste género é adicionado no texto a expressão regular escrita em Perl que caracteriza a regra, assim como uma vírgula no final, de forma a que possam ser adicionadas mais expressões regulares na lista criada. Esta regra é também válida para o último elemento desta lista a ser inserido, dado que na linguagem Perl essa vírgula é aceite.

Após a transformação das linhas do ficheiro de configuração iniciadas por “P:”, e assumindo que se segue um bloco de código, dado que, mesmo que não exista no ficheiro de configuração, ele é introduzido um bloco de código que não tem qualquer acção, é necessário fechar a lista anterior, assim como adicionar uma nova entrada na estrutura que representa uma regra com chave igual a “codigo”. Esse código é inserido dentro de uma função, em que tem como parâmetro de entrada um array associativo, com os valores das variáveis criadas no ficheiro de configuração, e que devolve o mesmo array associativo, de forma a poder ser utilizado como variáveis nas respostas configuradas. É também feita a conversão da sintaxe definida para blocos de código no ficheiro de configuração de módulos para sintaxe Perl, de forma a se poder executar as acções definidas neste bloco quando a Etelvina reconhece uma mensagem que despolete a execução deste bloco. Essa transformação consiste na substituição de todas as expressões que se iniciam com a sequência \$\$ para apontarem para um valor do array associativo que entra como parâmetro na função. Esta transformação está demonstrada na tabela 3.1.

Como o bloco e código se encontra todo numa só linha de texto, dado que previamente são removidos todos os caracteres “\n”, tal como fora explicado anteri-

ormente, cria-se nesta fase a lista para as possibilidades de resposta, num processo semelhante à criação da lista com as mensagens de entrada que devem ser reconhecidas. No entanto, o bloco de respostas possui diferenças comparativamente ao bloco de mensagens de entrada.

Enquanto que, ao tratar uma linha do bloco de mensagens de entrada, apenas se tem de adicionar os seus valores na lista correspondente, uma linha do bloco de respostas possui comportamentos diferentes, que se encaixam em três tipos possíveis:

1. Ser uma resposta seguida de outra resposta;
2. Ser uma resposta seguida de uma nova regra;
3. Ser uma resposta em que representa a última linha do ficheiro.

Perante as possibilidades existentes, é necessário criar as regras de reescrita, de forma a se conseguir reescrever as regras para todos os pontos. Assim, para o ponto 1 e 3 da lista anterior, elaboraram-se regras semelhantes às criadas para as mensagens de entrada a nível de inserção dos valores na lista correspondente, em que, uma vez que a lista já se encontra aberta, basta adicionar as respostas seguidas de vírgulas. Ao contrário do comportamento definido para as mensagens de entrada, não é necessário criar expressões regulares, uma vez que não é necessário fazer testes às expressões, pois a mensagem que está a ser tratada é a que a EteLVina devolverá ao utilizador. Nesta fase ainda não é feita a substituição das variáveis definidas pelo utilizador, dado que apenas no momento da resposta, em que as variáveis já têm os seus valores definidos, é que faz sentido efectuar essa substituição.

Para o ponto 2 é necessário, além do tratamento existente nos outros dois, ter ainda em atenção que é necessário fechar a lista correspondente às respostas da regra que está a ser tratada, além de iniciar a regra seguinte, à semelhança do tratamento que é efectuado inicialmente. Assim garante-se a definição correcta das várias regras para a estrutura definida para armazenamento de tais dados.

Utilizando o exemplo 3.3, e aplicando as regras de reescrita, o texto que resulta é o visível no exemplo 3.8.

Exemplo 3.8: Exemplo de configuração de módulo após aplicação das regras de reescrita.

```
$entrada=[{
  perguntas => [
    qr/Qual\s+a\s+temperatura\s+em\s+(?<localidade>teste)\?/i,
    qr/Em\s+(?<localidade>\s*\w+(\s*\w+)*),\s+quantos\s+graus\s+est[ãa]o\?/i,
  ],
  codigo => sub { my $hash = shift;
                $valor = temperatura($localidade);
                return $hash;},
  respostas => [ "Em <localidade> estão <valor>°C.",
```

```
"Estão <valor>°C em <localidade>.",
"<localidade>:<valor>." ] ] ] ;
```

Como é possível verificar, é construída uma lista, em que os seus valores são arrays associativos que armazenam as expressões regulares que definem as mensagens de entrada que a Etelvina reconhece, assim como o código que é executado quando uma mensagem de entrada da regra é enviada, e por fim a lista de respostas possíveis para a regra em questão.

Após a transformação das regras em texto com sintaxe Perl, resta executar esse texto, de modo a se obter a regra em questão na estrutura de dados definida.

De forma a otimizar o carregamento dos módulos, foi criada uma estrutura que suporta as regras definidas para cada módulo. Esta estrutura é um array associativo, em que a chave é o nome do módulo configurado e o seu valor a lista de regras definidas para o módulo em questão. No entanto, como não é obrigatório ter um módulo Perl configurado (tal como explicado na secção 3.1.1), é necessário utilizar a chave gerada como nome do módulo para armazenar a estrutura com as regras em questão.

O carregamento dos módulos é feito utilizando a metodologia *lazy-load*. Esta metodologia consiste na inicialização de um objecto apenas quando é necessário esse mesmo objecto, e foi utilizado no projecto para transformar as regras de um módulo apenas quando é invocado pela primeira vez esse mesmo módulo.

Desvantagens	Vantagens
<ul style="list-style-type: none"> •Ao se carregar os módulos uma só vez, se for efectuada alguma alteração tempo de execução no ficheiro de configuração, não será carregada a alteração sem ser executada alguma acção para forçar esse carregamento •Possuindo um grande número de módulos configurados, com muitas regras, a memória alocada será superior em relação a se fossem efectuadas sempre as transformações 	<ul style="list-style-type: none"> •É apenas necessário efectuar a transformação dos ficheiros uma única vez, evitando assim processamento constante e acessos de leitura do disco •Apenas são carregados os módulos que já foram utilizados para testar se fazem <i>match</i> às mensagens de entrada

Tabela 3.3: Vantagens e desvantagens da utilização da metodologia utilizada para carregamento de módulos.

Tendo em conta que é necessário respeitar a ordem de carregamento dos módulos definidos no ficheiro de configuração do motor (ver secção 3.1.1), o algoritmo utilizado para esse carregamento inicia-se com uma mensagem enviada à Etelvina por um

utilizador. Executando um ciclo sobre a lista de módulos configurada, e respeitando a ordem, a aplicação testa se a mensagem faz *match* a alguma expressão regular definida para o módulo em questão. Ao efectuar esse teste, é feita a transformação do ficheiro de configuração na estrutura definida, caso seja a primeira vez que o módulo é testado. Caso não obtenha resposta, executa os mesmos passos para todos os módulos configurados, assegurando assim o carregamento de todos eles, executando o fluxo demonstrado na figura 3.5.

3.1.4 Exemplos de módulos criados para demonstrar usabilidade da Etelevina

Módulo de meteorologia

O módulo de meteorologia tem como grande objectivo dar respostas sobre informações que não são fixas, pois o estado meteorológico, como se sabe, é variável pelos espaços temporal e físico. Assim sendo, a resposta a dar pela **Etelevina** será com base numa consulta a um serviço online sobre o estado meteorológico de uma determinada localidade.

Para invocar este módulo, tem de se enviar um parâmetro: a localidade. As funções disponibilizadas são a `condicaoMeteorologica` e `temperatura`, em que ambas recebem o local sobre o qual se deseja obter essas informações.

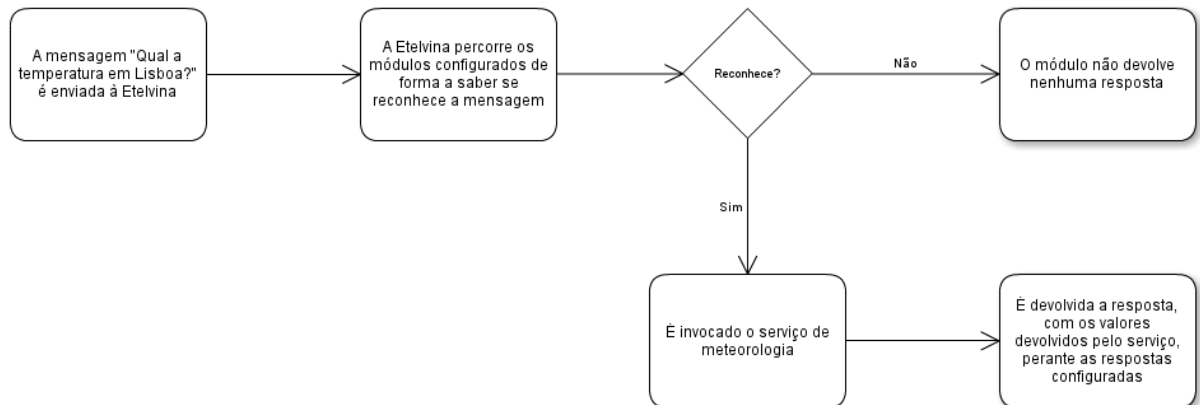


Figura 3.6: Esquema representativo do fluxo de uma mensagem no módulo de meteorologia.

A necessidade da criação de um módulo sobre o estado meteorológico surgiu após uma análise a variadas conversas entre humanos. Nessa análise percebeu-se que a grande maioria das conversas, exceptuando as que são exclusivas a um determinado tema, envolvem sempre o estado meteorológico. Expressões como “hoje está frio” ou “chove que se farta” são utilizadas diariamente pela maioria das pessoas, além de que quando não se tem um tema de conversa com alguém, acaba-se sempre por falar sobre o tempo. Assim sendo, faz sentido que a Etelevina converse sobre

o estado meteorológico, apesar de apenas dizer qual a temperatura e estado do tempo nas localidades pretendidas. No entanto, tendo em conta a possibilidade de configurações, a quantidade de respostas sobre algo apenas com os valores do estado do tempo e temperatura pode ser bastante diversificado.

Módulo de cálculo

O módulo de cálculo surge de forma a conseguir calcular todas as expressões que façam *match* à gramática 3.1. Apesar de nos blocos de código dos ficheiros de configuração de um módulo ser possível fazer cálculos elementares sem se importar nenhum módulo, existem funções na gramática definida para expressões matemáticas que não estão disponíveis sem essa importação. Assim, de forma a garantir que todas as expressões que sejam configuradas como **MATEXP** consigam devolver o resultado calculado, foi criado um módulo com uma única função, com o nome `doCalculo`, em que tem como parâmetro a expressão regular matemática, e devolve o resultado dessa expressão.

3.2 Conversas multi-utilizador

Tendo em conta todas as potencialidades que os clientes de mensagens instantâneas oferecem, uma das questões que surgiram como possível limitação da Etelvina prende-se com o facto de não ter suporte para conversas multi-utilizador.

As conversas multi-utilizador têm um paradigma bastante distinto das conversas entre dois utilizadores. Antes de mais, a conversa com o tipo acção-reação deixa de ser possível, uma vez que a conversa não é entre apenas dois intervenientes. Isto porque as mensagens em grupos de conversação não têm como base a conversa directa entre dois intervenientes.

Uma das principais características que uma conversa deste género tem é a existência de mensagens que possuem características que identificam o destinatário de uma mensagem. Esse mecanismo seria fácil de implementar caso essa característica fosse o nome, sendo apenas necessário fazer *match* a mensagens que contivessem o nome definido no ficheiro de configuração do motor, de forma a devolver uma resposta. No entanto, outro tipo de característica seria muito complicado de implementar mantendo as regras definidas para conversas entre apenas dois intervenientes, tais como invocar alguém pelo seu sexo, idade ou descrição física, características essas muito utilizadas em conversas entre vários intervenientes humanos.

Outra das características que deve ter uma conversa deste género é o triangular a conversa, de forma que não se prenda a uma conversa entre dois intervenientes com os outros intervenientes a assistirem. Assim, seria necessário que a Etelvina conseguisse receber uma mensagem e depois responder de forma a que fosse invocado outro interveniente.

Assim sendo, e por falta de tempo de implementação, pois era necessário criar um novo modelo distinto do implementado para conversas entre dois intervenientes, a Etelvina não possui suporte para este tipo de conversas.

3.3 Estrutura de comunicações

A implementação da Etelvina baseia-se na comunicação através de um cliente de *Instant Messaging*, tendo sido escolhido neste caso o cliente *Google Talk*, um cliente assente no protocolo de comunicação XMPP. No entanto, e tal como supracitado, a possibilidade de utilização de um novo cliente de mensagens instantâneas, e caso o seu protocolo de comunicação não seja o XMPP, tal como o *Messenger*, que utiliza o *Microsoft Notification Protocol* (MSNP) sobre TCP, não foi descartado, bastando alterar as funções de autenticação no servidor, de envio e recepção de mensagens e, por fim, as de controlo de presença de contactos para se obter a Etelvina num cliente de mensagens instantâneas que não siga o protocolo XMPP. No caso do cliente da *Microsoft*, o *Messenger*, foi estudada a API `Net::Msmgr`, o que permitiu verificar que as alterações referidas seriam suficientes para a utilização do protocolo utilizado por este cliente de *instant messaging*.

3.3.1 Protocolo XMPP

O *Extensible Messaging and Presence Protocol* ou XMPP é um protocolo aberto de *Instant Messaging*, especificado pela *Jabber*[Jab09], onde qualquer pessoa que tenha domínio e uma ligação à Internet pode executar o seu próprio servidor XMPP e conversar com utilizadores noutros servidores.

No projecto optou-se por desenvolver a Etelvina para a conversação através do servidor XMPP da *Google*, dado a sua popularidade e facilmente ser possível iniciar uma conversação através do cliente *Google Talk* ou das suas versões online.

O *Google Talk* é um cliente de mensagens instantâneas da *Google* que surgiu em 2005, onde apenas tinha suporte à conversação textual. Tornou-se bastante popular devido à sua integração no *Gmail*, pois o número de contas existentes para e-mail era grande, e daí a sua penetração no segmento de clientes de mensagens instantâneas foi rápida.

Voltando ao protocolo, o XMPP é inspirado em XML e toda a informação que suporta é baseada no *streaming* de XML. O protocolo foi construído de forma a ser expansível, o que permitiu, associado às capacidades inerentes ao uso do XML, um crescimento a nível de funcionalidades, tais como a utilização de VoIP. Aliás, a “extensibilidade é uma das maiores vantagens do XMPP”[GTa10].

3.3.2 API de comunicações utilizada - Net::XMPP

De forma a poder utilizar um servidor XMPP, foi necessário utilizar uma API que suportasse este protocolo. A API utilizada é a `Net::XMPP[Eat10]`, um módulo para Perl que suporta as mais variadas funcionalidades, desde o envio/recepção de mensagens, assim como o controlo de presença/ausência dos contactos.

A utilização da API torna-se bastante fácil após saber como são processadas as mensagens, sendo apenas necessário para realizar a conexão o nome do servidor, porta, o tipo de conexão e se é pretendido usar TLS (*Transport Layer Security*).

Exemplo 3.9: Utilização da API `Net::XMPP` para conexão ao servidor de mensagens instantâneas.

```
my $Connection = new Net::XMPP::Client();
my $status = $Connection->Connect(
    hostname => 'talk.google.com',
    port => 5222,
    componentname => 'gmail.com'
    connectionType => 'tcpip',
    tls => 1);
```

Após a conexão ao servidor estar estabelecida, resta fazer a autenticação e dizer quais as funções que irão fazer o tratamento da ligação.

Exemplo 3.10: Utilização da API `Net::XMPP` para autenticação no servidor de mensagens instantâneas e tratamento de eventos de mensagens e presença.

```
my @result = $Connection->AuthSend(
    username => "utilizador",
    password => "*****",
    resource => "gmail.com"
);

$Connection->SetCallbacks(
    send => \&endCallBack,
    receive => \&receiveCallBack,
    message => \&messageCallBack,
    presence => \&presenceCallBack,
);
```

Neste momento, todas as mensagens recebidas despoletam a função com o nome *messageCallBack*, podendo assim ser possível analisar as mensagens recebidas, pois a função tem dois parâmetros: um identificador da mensagem e um objecto do tipo `Net::XMPP::Message`.

Ao ser despoletada a acção de análise da mensagem, é garantida a resposta a essa mensagem, mesmo que entretanto a Etelvina receba outras mensagens de outros utilizadores, logo não é possível uma mensagem analisada ser respondida para outro utilizador que não seja o que enviou essa mensagem.

3.4 Etelvina vs AIML

Tendo em conta a linguagem definida para a configuração da Etelvina após a análise de uma já existente (AIML), é importante fazer comparações e verificar a verdadeira necessidade da definição de uma nova linguagem.

Considere-se como Etelvina toda a estrutura montada e explicada nas secções anteriores, assumindo que se trata de uma linguagem, de forma a poder comparar equitativamente ambas as tecnologias. Assim, a principal semelhança entre as duas linguagens é que ambas existem para criar Elizas, e que são configuráveis quer por módulos, no caso da Etelvina, quer por categorias, no caso do AIML. Ambos reagem através de um sistema de *match* de padrões, e perante esses padrões, devolvem uma resposta ao utilizador após uma mensagem deste.

No entanto, as diferenças entre as duas tecnologias são bastantes. Antes de mais, a plataforma para o qual são desenvolvidas é completamente diferente: enquanto a Etelvina foi construída de forma a se apresentar como um contacto de um servidor de mensagens instantâneas, tais como o *Google Talk*, o *Windows Live Messenger*, o *Yahoo Messenger* ou outro qualquer do género (desde que fornecido o suporte para o cliente desejado), a linguagem AIML foi criada para as Elizas serem executadas ou em modo batch, ou numa página *web*.

A Etelvina foi desenhada para ser um contacto de um cliente de mensagens instantâneas para ser um contacto útil para o utilizador, tal como supracitado, e, tendo em conta a utilização deste género de serviços, é relativamente simples de utilizar em qualquer lugar, dada a frequência de uso dos utilizadores dos seus clientes de mensagens instantâneas, enquanto se necessitassem de abrir um sítio na *Internet* ou executar um programa batch, torna-se uma ferramenta mais específica e o utilizador só irá utilizar a Eliza esporadicamente.

A nível de configuração, ambas as tecnologias permitem computação no tratamento dos comportamentos definidos, de forma a se poder obter resultados mais complexos. No entanto, as duas tecnologias divergem entre si na forma como é possível executar essa computação. Em AIML estão definidas dois elementos para serem executados em processadores externos. Esses dois elementos são representados pelas etiquetas *<system>* e *javascript*, e ambas não possuem quaisquer atributos.

Exemplo 3.11: Exemplo da utilização da etiqueta *<system>* em AIML, utilizando os comandos *echo* e *date +%H* disponível num ambiente UNIX, e que devolve a hora actual.

```
<category>
  <pattern>QUE HORAS SAO</pattern>
  <template>
    <system>echo "São 'date +%H' horas"</system>
  </template>
</category>
```

A etiqueta *<system>* instrui o interpretador de AIML para enviar o seu conteúdo, com o pré-processamento adequado, para o interpretador de comandos do

sistema da máquina, máquina essa que está a executar o interpretador de AIML. O pré-processamento necessário consiste na substituição de dados da personagem AIML guardados nas respectivas variáveis, assim como no tratamento de elementos *template*.

Exemplo 3.12: Exemplo da utilização da etiqueta `<system>` em AIML, em que é necessário efectuar um pré-processamento, para obter o nome da personagem construída.

```
<category>
  <pattern>QUAL O TEU NOME</pattern>
  <template>
    <system>echo '<get name="name"/>'</system>
  </template>
</category>
```

A etiqueta `<javascript>` é semelhante à instrução anterior, com diferença de, em vez de serem enviados comandos do sistema, são enviadas instruções na linguagem *JavaScript* para um interpretador dessa linguagem que está na máquina onde que executa o interpretador AIML.

Exemplo 3.13: Exemplo da utilização da etiqueta `<javascript>` em AIML para o cálculo da raiz quadrada de um número.

```
<category>
  <pattern>CALCULA RAIZ QUADRADA *</pattern><template>
    <think>
      <set name="numero"><star/></set>
      <set name="err">Huh? Isso não é um número</set>
    </think>
    A raiz quadrada de <get name="numero" /> é
    <script language="JavaScript">
      var x = '<get name="numero"/>';
      x = x.replace(' point ', '.');
      x = parseFloat(x);
      if( isNaN(x) ) {
        var r = '<get name="err" />';
      } else {
        var r = Math.sqrt(x);
      }
      document.write(r);
    </script>
  </template></category>
```

Na linguagem criada para a configuração de módulos da Etelvina é também possível a existência de computação. Essa computação é definida no ficheiro de configuração dos módulos, sendo executada caso exista o bloco de código, bloco esse caracterizado pelos caracteres *C:*, na regra que tem definida no bloco de perguntas uma frase que faça *match* à mensagem de entrada recebida. Este bloco de código

é escrito numa linguagem muito semelhante à linguagem *Perl* e que apenas difere desta linguagem em alguns aspectos, tais como a utilização de variáveis declaradas com os símbolos $\$$, que indicam que a variável em questão foi definida no bloco de perguntas que antecede o bloco de código a executar, ou que é necessária no bloco de respostas da regra em questão, tal como foi explicado com mais pormenor na secção 3.1.2.

Exemplo 3.14: Exemplo da configuração da raiz quadrada de um número na Etelvina.

```
P: Calcula raíz quadrada <matExp,MATEXP>
C: valor = sqrt(doCalculo(matExp));
R: A raíz quadrada de <matExp> é <valor>.
```

Comparando os exemplos 3.13 e 3.14, pode-se verificar que a computação executada pela Etelvina não executa nenhum teste para verificar se o número de entrada é um número, pois a pergunta está configurada para apenas aceitar como entrada o texto “Calcula raíz quadrada ” seguida de uma expressão matemática, o que permite que existam cálculos de várias funções antes de ser calculada a raiz do número em causa.

À semelhança da computação permitida em AIML, toda a computação dos blocos de código é executada na máquina onde a Etelvina está a ser executada.

Tanto em AIML, como na Etelvina, é permitida a existência de mais do que uma mensagem de entrada para um determinado comportamento. Assim, não é necessário redefinir as regras totalmente para ter mais uma entrada que seja aceite. No entanto, e dadas as diferenças entre as metodologias de configuração que cada tecnologia possuem, as formas de acrescentar mais uma mensagem de entrada para um comportamento são distintas. O mesmo cenário ocorre nas mensagens de saída, em que podem ser sorteadas de um conjunto de respostas.

Em AIML, para se definir várias mensagens de entrada para um determinado comportamento é necessário recorrer à etiqueta *<srail>*.

Exemplo 3.15: Exemplo da utilização da etiqueta *<srail>* e *<random>* em AIML.

```
<category>
  <pattern>ADEUS</pattern>
  <template>
    <random>
      <li>Adeus!</li>
      <li>Goodbye!</li>
      <li>Adios, adieu, auf wiedersehen, goodbye!</li>
    </random>
  </template>
</category>
<category>
  <pattern>BYE BYE</pattern>
  <template>
```

```
        <srai>ADEUS </srai>
    </template>
</category>
<category>
    <pattern>TCHAU </pattern>
    <template>
        <srai>ADEUS </srai>
    </template>
</category>
```

Com esta etiqueta, consegue-se a reutilização da mesma regra de resposta para várias mensagens de entrada. No entanto, é necessário criar a categoria e depois atribuir-lhe o template que se pretende, tal como é visível no exemplo 3.15.

Quanto às mensagens de saída, em AIML é necessário utilizar a etiqueta *<random>* para definir que a resposta será sorteada entre as várias definidas, que se encontram dentro das etiquetas **.

Na Etelvina, este processamento também existe, no entanto a linguagem permite a definição de várias mensagens de entrada para a mesma regra. Utilizando como base as mensagens definidas no exemplo 3.15, que são “Adeus”, “Bye bye” e “Tchau”, e “Adeus!” como mensagens de entrada e “Adeus!”, “Goodbye!” e “Adios, adieu, auf wiedersehen, goodbye!” como mensagens de saída, a configuração na Etelvina para o mesmo comportamento é a presente no exemplo 3.16.

Exemplo 3.16: Exemplo de utilização, na Etelvina, de várias mensagens de entrada e saída numa só regra.

```
P: Adeus
P: Bye bye
P: Tchau
R: Adeus!
R: Goobye!
R: Adios, adieu, auf wiedersehen, goodbye!
```

O histórico de conversa existente também diferem nas duas tecnologias. Em AIML, apenas a última frase do utilizador é armazenada, de forma a poder ser utilizado na definição de novas regras através da etiqueta *<that>*. Na Etelvina, o número de mensagens recebidas a ser armazenado é configurável, pelo que pode ser o quanto se deseje, desde que não haja limitações de memória. No entanto, essas mensagens, para serem úteis na definição de regras, implica que se tenha de utilizar o bloco de código a executar para efectuar as comparações desejadas. Assim, o uso destas mensagens não é tão intuitivo e simples como na linguagem AIML.

Por fim, a comunidade de utilizadores/programadores das duas tecnologias são muito diferentes. Enquanto o AIML possui uma comunidade vasta de utilizadores,

	AIML	Etelvina
<i>Plataforma na qual é executável</i>	É executado num programa de computador ou numa página web.	É executado como um contacto de um cliente de mensagens instantâneas.
<i>Executa computação?</i>	Sim, através de funções do sistema no qual está ser executado, ou através de <i>JavaScript</i> .	Sim, através de uma linguagem criada que tem como base o <i>Perl</i> .
<i>Possibilidade de adicionar mais mensagens de entrada e saída para uma determinada regra?</i>	Sim, adicionando, para mensagens de entrada, uma nova categoria e associando-a através da etiqueta <i><srain></i> e associando o mesmo padrão, e, para mensagens de saída, através da etiqueta <i><random></i> para sortear uma hipótese de resposta definida pela etiqueta <i></i> .	Sim, sendo necessário acrescentar no ficheiro de configuração uma nova linha iniciada com o texto "P:" para definir uma nova mensagem de entrada aceite, e uma nova linha iniciada com o texto "R:" para adicionar uma hipótese de resposta.
<i>Histórico de conversa armazenado</i>	Apenas possui a última mensagem escrita pelo utilizador.	Número configurável pelo utilizador.
<i>Comunidade de utilizadores</i>	Comunidade vasta, com alguns projectos a terem recebido o prémio Loebner.	Comunidade inexistente.

Tabela 3.4: Tabela resumo de comparação entre a estrutura adoptada para a Etelvina e o AIML.

o que permite que o número de *templates* existentes nesta linguagem seja grande, tendo Elizas escritas em AIML ganho, por diversas vezes, o prémio Loebner, a Etelvina ainda não possui essa comunidade, o que a limita na quantidade de regras definidas.

Capítulo 4

Base de conhecimento da Etelvina

Sendo a Etelvina uma personagem para ser utilizada como um contacto de clientes de mensagens instantâneas, serão variadas as perguntas que lhe serão colocadas, em que, para poder responder a essas perguntas, é necessário ter um modo de obter as respostas. Para a obtenção dessas respostas poderão ser utilizados, em tempo de execução, serviços online para a obtenção da informação necessária para responder às questões colocadas pelos utilizadores.

No entanto, a utilização de serviços online em tempo de execução leva a que as respostas demorem muito tempo para serem obtidas, assim como, caso o número de conversas em simultâneo que utilizem esses serviços seja elevado, que existam atrasos nas variadas conversas. Deste modo, a alternativa aqui representada foi a construção de uma base de dados local de conhecimento, sendo alimentada por serviços online, com o objectivo de obter as informações com o mínimo número de acessos remotos possível.

Nesse sentido, desenvolveu-se um protótipo de uma ferramenta de extracção de conhecimento de informação através da Wikipédia, conhecimento que é armazenado numa base de dados de suporte às informações obtidas nessa enciclopédia online.

O capítulo encontra-se dividido em duas partes, a primeira que retrata alguns algoritmos existentes para a extracção de conhecimento, como um estado da arte, e a segunda sobre a construção da base de conhecimento elaborada para este projecto.

4.1 Algoritmos de obtenção de conhecimento

Após uma análise prolongada sobre qual algoritmo de obtenção de conhecimento deveria ser implementado, as alternativas que ficaram para discussão foram as seguintes:

1. Implementação de um sistema de *Question Answering* baseado em classificação de questões;

2. Implementação de um sistema de extracção de informações/factos para determinados temas.

Utilizando técnicas de *Information Retrieval*, que é uma área da computação que lida com o armazenamento de documentos e a recuperação automática de informação associada, foi iniciada a implementação de um sistema de classificação de perguntas *Question Answering*.

Esse sistema de classificação de perguntas pretendia classificar as perguntas de acordo com o tipo de resposta, ou seja:

- Local - quando se pretende que a resposta seja um local, para perguntas como “Onde fica a Torre de Belém?”;
- Pessoas - para informações sobre pessoas, em perguntas como “Quem foi Fernando Pessoa”;
- Organizações - quando se pretende informações sobre organizações, em perguntas do tipo “O que é a ONU?”;
- Medidas - quando a informação é numérica, em perguntas tais como “Qual o número de modalidades olímpicas?” ou “Qual a altura da Torre dos Clérigos?”;
- Tempo - quando a informação é temporal, como nas questões “Qual a idade da Anna Kournikova?” ou “Quando foi a Revolução dos Cravos?”;
- Outras, pois existe um sem número de diferentes tipos de resposta. Por exemplo, listas, como a enumeração dos membros de um governo, ou das línguas faladas em determinado país.

Ao receber uma pergunta, estes sistemas devem classificá-la numa das categorias suportadas. Esta classificação é útil para que se possa utilizar técnicas diferentes na obtenção da informação sobre cada uma delas. Por exemplo, se a pergunta pertence à categoria de medidas, sabe-se que a resposta deve conter um valor numérico.

Utilizando o texto livre de uma entrada da Wikipédia, pensou-se em utilizar uma técnica semelhante à utilizada nos artigos *Priberam's Question Answering System in QA@CLEF 2007*[ACF⁺07] e *Priberam's Question Answering System for Portuguese*[AFM⁺05]. A técnica consistia em indexar previamente artigos da Wikipédia como texto livre de marcações, fazendo algumas pequenas adaptações, utilizando estratégias sofisticadas capazes de extrair conhecimento a partir do texto. Esta indexação consiste num processo, através do qual um conjunto de documentos alvo é analisado, com o objectivo de recolher informações para o arquivo de indexação. Assim, para cada documento, são recolhidas os domínios ontológicos e terminológico mais relevantes e, para cada frase, a categoria da pergunta para que contém respostas possíveis. Aproveitando o título do artigo e o seu link, indexavam como

linguagem natural o texto existente numa entrada e criaram um sistema de resolução anafórica para a Wikipédia, pois muitas das anáforas¹ presentes num artigo são o título do artigo, sistema que consistia na substituição, durante o *parsing* do texto, de todas as frases que não tivessem sujeito ou que o sujeito fosse um pronome pessoal pelo título do artigo, de forma a obter uma frase estruturada sobre o tema.

Ao receber uma questão, o sistema analisa-a e interpreta-a, iniciando o processo de interpretação pela atribuição de categorias para a questão, através de padrões de pergunta, que permitem a atribuição de várias categorias a uma pergunta, evitando assim dificuldades em atribuir a categoria mais adequada. O passo seguinte de análise da questão consiste na extracção de elementos chave na questão, que podem ser palavras, frases, datas, números, abreviaturas, entre outras.

Após a análise da questão, são feitas consultas ao arquivo de indexação, utilizando os elementos chave anteriormente extraídos, calculando um peso para cada palavra, segundo regras de valorização, que têm em conta qual o tipo do elemento chave, atribuindo um peso também aos artigos indexados perante a pergunta feita. Após essas atribuições, são tidos em conta os trinta artigos com o maior peso perante a questão, que são analisados com mais pormenor.

Por fim, são seleccionadas, através de um processo de extracção de respostas, as frases que, presumivelmente, contêm a resposta para a pergunta feita, atribuindo-lhes também pesos. A frase que tiver um peso maior é a que é respondida.

Mas a adopção de uma técnica deste género foi abandonada pelos motivos que se seguem.

No *Question Answering*, é particularmente importante alcançar uma alta precisão em analisar as questões [Her01]. Se a análise de uma palavra da questão não for bem sucedida, a resposta apenas poderá ser bem respondida caso as outras frases consigam ser suficientes para a produção da resposta. No entanto, se a questão, na sua totalidade é analisada de forma errada, o fracasso da resposta é garantido.

Logo, para ter um sistema deste género, a análise das perguntas teria de ter uma percentagem de sucesso satisfatória. Mas, tendo em conta a necessidade de implementar o sistema num espaço temporal curto, teria de se proceder a simplificações, o que iria fazer com que a análise das perguntas fosse um pouco primitiva, tendo como consequência a obtenção de baixas percentagens na avaliação das perguntas.

O primeiro dos problemas era o número de perguntas que cada um dos grupos acima referidos iria conter. Enquanto o grupo “Tempo” teria um número reduzido de questões a avaliar (“quando?” ou “em que tempo?”), o grupo “Medida” teria um número indeterminado de questões. Para dar ideia do número de questões referentes a este grupo, basta pensar nas medidas e unidades de peso, distância, largura, entre outras (além da possível necessidade de conversões entre unidades para a obtenção de uma resposta favorável).

¹Uma anáfora consiste na repetição deliberada da mesma palavra no começo de várias frases.

Outro dos problemas patentes nesse tipo de implementação era a quantidade de perguntas que não iriam ter uma classificação concreta. Isso implicava que o número de perguntas que iriam ficar sem um tratamento específico seria excessivo, pois a categoria é muito vasta e seria impossível conseguir responder a uma grande parte das perguntas.

Dois dos tipos de perguntas mais utilizados (“Como?” e “Porquê?”) não podem simplesmente ser mapeados para um dos tipos de pergunta, tal como refere o artigo *A question answering system supported by information extraction* [SL00], o que leva à análise do resto da pergunta para que o sistema consiga detectar com que tipo de pergunta se depara.

Dada a complexidade deste trabalho e o curto período para a sua implementação (já que o objectivo principal desta dissertação é a construção de um motor de Etelevina e não a construção de um sistema de resposta a perguntas), optou-se por uma simplificação drástica. O sistema implementado consiste na extracção de informação (doravante designado de IE, do inglês *Information extraction*).

4.2 Construção de uma Base de Conhecimento

No âmbito do processamento de linguagem natural, entende-se como IE um tipo de *Information Retrieval* que consiste na extracção automática de informação sobre documentos semi-estruturados. Assim sendo, depois de ser seleccionada uma ou mais fontes de dados de onde obter a informação, basta a análise (ou parsing) do documento, e a conversão dos dados obtidos numa estrutura que permita posteriormente a fácil utilização desses dados.

De forma a se conseguir uma fonte de informação aberta, com uma grande diversidade de informações e acima de tudo informação actualizada constantemente, optou-se por utilizar a Wikipédia como fonte de dados. A utilização desta enciclopédia virtual satisfaz requisitos de ter uma diversidade de informações actuais, apresentando como grande problema a volatilidade da informação. Isto acontece devido à forma como a enciclopédia é construída. Ao poder ser facilmente adicionada informação sobre um determinado tema, nada garante que a informação presente na entrada seja verdadeira. Além de que pode, eventualmente, acontecer uma alteração de uma entrada que contenha informação fidedigna para passar para informação completamente errada.

Pesando todos os prós e contras da utilização da Wikipédia, decidiu-se que seria uma boa fonte de dados, uma vez que dificilmente se conseguiria encontrar uma fonte de dados que fosse satisfatória para o problema em questão. Quanto à informação falsa, após uma análise do que seria possível para a evitar, decidiu-se que caso seja consumida apenas dará uma faceta mais “humana” à Etelevina.

A informação apenas é extraída no momento em que é enviada à Etelevina uma mensagem com o termo a pesquisar. Depois de extraída da informação, e após o

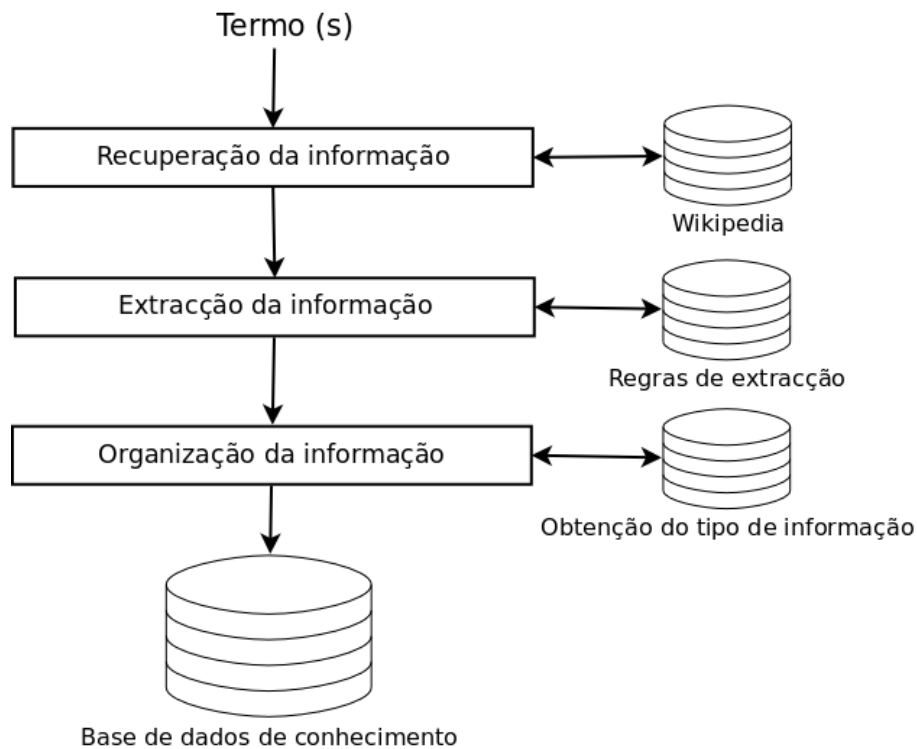


Figura 4.1: Diagrama que mostra o fluxo de aquisição de conhecimento perante um termo.

seu tratamento, a informação é armazenada numa base de dados, de forma que da próxima vez que for necessário dar informação sobre esse mesmo termo, não seja necessário recorrer à Wikipédia para retornar o resultado, devolvendo a informação que ficou armazenada anteriormente na base de dados.

A base de dados em questão é muito simples, e bastante aberta de forma a que o utilizador tenha a liberdade de acrescentar tabelas na base de dados à sua vontade, de forma a se conseguir responder às necessidades de armazenamento de informações que se tenha.

Assim, a base de dados da Etelvina só possui uma tabela obrigatória, de seu nome “Conceitos”. Esta tabela foi construída de forma a ser simples de obter informações a partir dela, assim como de gravar informações simples que sejam extraídos de alguma fonte de dados, seja uma enciclopédia online ou através de mensagens de um utilizador.

A tabela conceitos possui os seguintes campos:

- **Chave;**
- **Conceito;**
- **Conceito2;**

- **Tipo.**

Como é possível verificar na listagem acima, a tabela é muito simples, dado o pequeno número de campos. No entanto, responde às necessidades de armazenamento da Etelvina, dado que se pretende apenas armazenar uma informação (campo “Conceito”), o seu significado (campo “Conceito2”) e qual o tipo ou categoria ao qual pertence (campo “Tipo”). A “Chave” permite a ligação entre a tabela de conceitos e as tabelas que sejam criadas pelo utilizador, conseguindo-se assim que, independentemente da complexidade do termo e do número de campos necessários para armazenar as informações que possui, o armazenamento de qualquer termo fique registado na tabela de “Conceitos”, assim como todas as informações que o constituem em tabelas que sejam criadas para o armazenamento dessas informações.

Apesar da quantidade de informação da Wikipédia ser enorme, a generalidade dos artigos existentes não se encontram estruturados, mas sim em texto livre. Isso impossibilita qualquer tentativa de obtenção de pequenas informações, estruturadas, passíveis de serem isoladas das restantes informações, e importantes sobre um determinado tema/assunto de uma entrada da enciclopédia. A hipótese de guardar todo o texto de uma entrada nunca foi uma opção para a obtenção de conhecimento da Etelvina, pois em vez de adquirir conhecimento específico, apenas seria uma forma diferente de apresentar uma entrada.

De forma a contornar esse problema, utilizaram-se pequenas informações que existem na Wikipédia sob o conceito de infoboxes.

4.2.1 Utilização de conhecimento obtido na Etelvina

De forma a extrair o conhecimento obtido pela Etelvina, foi necessário criar um mecanismo de interpretação sobre que termos, numa conversa regular, o que permite que essa informação seja adquirida e posteriormente utilizada sem problemas de maior se pretende obter mais informação. Um dos mecanismos pensados foi, para cada frase que fosse enviada para a Etelvina, pesquisar cada palavra na base de dados de conhecimento, e, caso não existisse o termo em questão, pesquisar na Wikipédia, de forma a saber que palavras pertencem a temas aos quais a Etelvina tem conhecimento e consegue processar a sua informação. Isso implicaria um número de pesquisas à base de dados igual ao número de palavras de uma frase, e, no pior dos casos, um igual número de consultas à Wikipédia para testar o tipo de palavra.

De forma a evitar inúmeras pesquisas à Wikipédia, resolveu-se inserir na base de dados uma lista de palavras pré-conhecidas e apenas fazer a sua associação com o tema a que se refere. Assim, com uma frase com termos conhecidos pela Etelvina, evita-se a utilização de recursos externos, bastando fazer uma pesquisa na base de dados pelo termo desejado. Para se conseguir manter as categorias, sobre as quais a Etelvina tem conhecimento, actualizadas, foram gerados métodos para actualizar a informação regularmente. Ou seja, a cada período de tempo, período esse configurá-

vel, a Etelvina irá actualizar as diversas listas de palavras dos temas sobre os quais tem conhecimento, evitando que a informação que tem não seja facilmente obsoleta.

Assim, numa frase composta por várias palavras, e analisando cada uma das palavras por ordem de aparecimento na frase, o mecanismo adoptado inicialmente era o seguinte:

1. SE:

Conhece a palavra guarda essa palavra numa lista e volta para o ponto 1. com a palavra seguinte

Não conhece a palavra volta para o passo 1. com a palavra seguinte

2. Se a lista de palavras não é vazia, verifica a lista gerada, verificando quais as palavras que conseguem devolver mais informação. Este teste implica saber quais os temas que a Etelvina tem mais conhecimento, e para fazê-lo foi necessário atribuir *ranks* a cada tema.
3. Depois de se saber qual a palavra com um maior *rank*, devolve-se a informação relacionada com essa palavra.
4. Caso a lista de palavras seja vazia, pesquisa informação na Wikipédia da última palavra analisada.

Quando a lista de palavras é vazia, optou-se por dar informação sobre a última palavra da frase, pois é geralmente a última palavra aquela que tem mais relevância.

No entanto, facilmente foram descobertas falhas que não permitiam respostas correctas à informação pretendida pelo utilizador. Isto porque nem sempre as palavras na língua portuguesa são “independentes”, isto é, por vezes só fazem sentido (pelo menos do ponto de vista humano) quando se associam a outras palavras. E quando se tratam de substantivos comuns, as falhas de apenas analisar palavra a palavra ainda mais visível se tornavam. Por exemplo, no caso de algumas localidades, como a localidade de *Vieira do Minho*, não devolveria qualquer tipo de informação, dado que existe, na Wikipédia uma lista de artigos que são ambíguos (no caso a região do *Minho* e o *rio Minho*), quando o que se pretendia era informação da localidade referida. Noutros casos, como *Macedo de Cavaleiros*, a informação que iria pesquisar era de *cavaleiros*, sujeitos que andam de cavalo. Logo a metodologia que foi seguida inicialmente rapidamente deixou de ser fiável.

Para tentar ser o mais fiável possível, tomou-se a decisão de analisar o maior número de palavras que uma freguesia portuguesa tenha no seu nome. Assim, e após uma pesquisa na *Lista de freguesias de Portugal*, descobriu-se que existem duas freguesias que têm no nome nada mais, nada menos, do que oito palavras. Essas freguesias são *Azinheira dos Barros e São Mamede do Sádão*, no concelho de *Grândola*, e *Nossa Senhora da Graça de Póvoa e Meadas*, no concelho de *Castelo de Vide*. Assim sendo, em vez de se efectuar uma pesquisa palavra a palavra, tem

de se fazer o mesmo de uma a oito palavras, que já é um número considerável para obter sucesso numa pesquisa. Assim, faz-se pesquisas sucessivas na base de dados com os termos que constituem as oito palavras, parando o ciclo quando o número de resultados obtidos na pesquisa à base de dados for igual a um e apenas um resultado.

Mesmo assim, esta restrição pode não ser suficiente, dado que existem termos iguais tanto dentro da mesma categoria, como em categorias diferentes. Exemplos disso são as freguesias portuguesas de *Cuba* e *França* e os países homógrafos, que criam uma ambiguidade que se torna complicado de gerir. Neste caso, a Etelevina deverá dizer as ocorrências que conhece, ignorando qual dos temas (freguesias ou países) na realidade se pretendia obter informação. Nesse caso, o número de ocorrências da base de dados será superior a um, mas os resultados obtidos correspondem exactamente ao termo pesquisado.

Para resolver este problema, decidiu-se atribuir um índice de relevância a cada termo, que corresponde à percentagem de palavras do termo que compõe a mensagem de entrada. A resposta cujo termo tenha a maior percentagem de relevância é o que será devolvido na resposta. Em caso de existirem relevâncias com o mesmo valor para várias respostas, será sorteada uma dessas respostas.

4.2.2 Infoboxes da Wikipédia

As infoboxes, ou caixas de informação, são caixas que contêm informação adicional sobre o artigo e têm como objectivo apoiar os mesmos. Geralmente são todos os tipos de caixas criadas com informação genérica, cronológica, bibliográfica, taxonómica, etc.

As infoboxes têm como principais trunfos a existência de estruturas fixas de informação, o que é fundamental para o sistema de IE, além de que possui um aglomerado de pequenas informações relativas a um determinado tema, sendo no entanto importantes. Isso faz com que se possa utilizar estas fontes de informação de forma a obter informações sobre um determinado tema.

As informações existentes numa infobox variam de acordo com o tema ao qual o artigo pertence. Isso criou a necessidade de se extrair apenas informação sobre determinados temas. Assim, o processamento para extrair informações sobre países, localidades de Portugal, ou de temas como personalidades é bastante distinto, dada a diferença de estrutura das infoboxes de cada um dos temas.

Para se conseguir extrair informação de uma infobox, teve de fazer-se inicialmente uma análise de quais os campos importantes sobre determinado tema. Assim, apenas se extrai a informação considerada útil (caso exista). Depois, foi necessário arranjar uma forma de conseguir transformar a estrutura de texto de uma infobox numa nova estrutura, de forma que se tenha apenas o valor correcto para cada chave.

No exemplo da infobox da figura 4.2, em que o seu código é o apresentado no exemplo 4.1, é possível verificar que o texto referente à infobox contém diversas informações que carecem de tratamento adicional, como por exemplo a sub-região a que

	
Brasão	Bandeira
Área	220,15 km²
População	14 724 hab. (2001)
Densidade populacional	67 hab./km²
N.º de freguesias	21
Fundação do município (ou foral)	1514
Região	Norte
Sub-região	Ave
Distrito	Braga
Antiga província	Minho
Orago	Nossa Senhora da Conceição
Feriado municipal	Segunda-feira subsequente ao primeiro Sábado de Outubro
Código postal	4850
Endereço dos Paços do Concelho	http://www.cm-vminho.pt
Sítio oficial	http://vieiradominho.blogspot.com
Endereço de correio electrónico	<i>Não disponível</i>

Municípios de Portugal 

Figura 4.2: Exemplo de infobox da Wikipédia.

o município pertence. A informação que é apresentada na Wikipédia é “Norte”, no entanto o código correspondente é “[[Região Norte (Portugal)|Norte]]”. Isto acontece por causa do sistema de *links* existentes na Wikipédia. Quando existem entradas na Wikipédia sobre um valor de uma qualquer chave da infobox, o valor dessa chave no código tem o seguinte formato: “[[*link_da_entrada*|valor]]”. Como no sistema de obtenção de conhecimento só interessa o valor, o *link* é ignorado.

Exemplo 4.1: Exemplo do código da infobox correspondente à figura 4.2 .

```

{{DadosMunicípioPortugal |
|município = Vieira do Minho
|imagem_brasão = VRM.png
|imagem_bandeira = Pt-vrm1.png
|gentílico =
|região = [[Região Norte (Portugal)|Norte]]

```

```

|subregião = [[Ave (subregião)|Ave]]
|distrito = [[Distrito de Braga|Braga]]
|província = [[Minho (província)|Minho]]
|área = 220,15
|população = 14 724
|data = 2001
|densidade = 67
|freguesias = 21
|fundação = [[1514]]
|orago = [[Maria|Nossa Senhora da Conceição]]
|feriado = [[Segunda-feira]] subsequente ao primeiro [[Sábado]]
           de [[Outubro]]
|codpostal = 4850
|url_cm = http://www.cm-vminho.pt
|url_oficial = http://vieiradominho.blogspot.com}}

```

Um dos maiores problemas que existem a fazer o mapeamento das informações numa infobox são os campos dinâmicos. Um desses casos diz respeito aos títulos de líderes que aparecem nas infoboxes dos países. Pelo facto dos cargos ocupados pelos líderes de um país variarem consoante o tipo de Governo, existem campos genéricos que têm como chave “título_líder” e portanto têm de ser tratados antes de serem inseridos na base de conhecimento da Etelvina. Por exemplo, em Portugal o tipo de governo é uma democracia parlamentar, com Presidente de República e Primeiro-Ministro. No Reino Unido, sendo o tipo de governo um sistema parlamentarista e monarquia constitucional, os seus líderes são a Rainha e o Primeiro-Ministro. Já a Suíça tem implantada uma República Federal, onde o líder é apenas o Presidente do Concelho. Isto faz com que as informações existentes numa infobox, mesmo que seja do mesmo tema, tenham variações em relação a nome e número.

Exemplo 4.2: Exemplo de código de campos variáveis no artigo “Portugal”.

```

|título_líder1 = [[Presidente da República Portuguesa|
                 Presidente]]
|nome_líder1   = [[Aníbal Cavaco Silva]]
|título_líder2 = [[Lista de primeiros-ministros de Portugal|
                 Primeiro Ministro]]
|nome_líder2   = [[José Sócrates]]

```

Para contornar esse problema, criou-se um sistema de mapeamento específico para casos semelhantes a este, que consiste no armazenamento de todas as informações que existam numa infobox. Posteriormente, é feita uma análise de quais os nomes dos campos que são terminados num algarismo, pois esse algarismo indica que existe mais do que um tipo de informação semelhante em que os verdadeiros nomes dos campos são variáveis. Vendo o exemplo 4.2, verifica-se que os nomes dos líderes de Portugal não se encontram correctamente associados. Logo, o pretendido é associar o valor do *nome_líder* ao valor *título_líder*. Para isso, cria-se um array

associativo, em que a chave corresponde ao verdadeiro nome do campo (por exemplo, na entrada “Portugal” do exemplo 4.2, a chave deve ser *título_líderm*) e o valor correspondente ao valor do campo (na entrada “Portugal” do exemplo 4.2, o valor deve ser *nome_líderm*).

A extracção de conhecimento de infoboxes permite, dada a especificidade dos campos de cada infobox por cada tema, a criação de tabelas na base de dados com informações específicas relacionadas com o tema em questão, perante cada entrada da infobox. Assim, a base de dados passa a possuir tem uma semântica que permite fazer pesquisas mais estruturadas sobre a informação que obteve para as tabelas específicas por tema criada, tendo assim a Etelvina uma capacidade de resposta a variações das questões sobre um determinado tema.

De forma a se poder extrair informações de infoboxes em que não existam mapeamentos específicos que correspondam ao tema da infobox em questão, criou-se um mecanismo de extracção genérica de informações. Este mecanismo consiste na extracção de todas as informações de infobox, num array associativo, em que a chave é o nome do campo da informação, e o valor corresponde ao texto atribuído ao campo. Para se utilizar essas informações nas respostas da Etelvina, recorre-se ao uso de *placeholders*, onde o texto a substituir tem o nome do campo, e é substituído pelo texto correspondente a essa chave. Com este mecanismo é possível assim ter extracções de informações de infoboxes genéricas, sendo apenas necessário conhecer os nomes das informações existentes em cada infobox, para utilização dos valores dessas informações nas respostas da Etelvina.

Limitações da técnica de extracção de conhecimento de infoboxes

Apesar da extracção de conhecimento das infoboxes ser um método em que se consegue obter bastantes informações, não é uma fonte de informação que permite acrescentar informações sobre novos temas na Etelvina. Isto porque sofre das seguintes limitações:

- **Necessidade de criação de um novo tipo de mapeamento por tipo de informação:** para se conseguir extrair informação de forma a ser possível utilizá-la numa resposta, implica que se tenha de criar manualmente um novo mapeamento para cada tipo de informação;
- **Apenas algumas entradas da Wikipédia têm infoboxes:** a maior parte das entradas não possuem qualquer infobox, pelo que utilizar a técnica de extracção de informação de uma infobox fica limitado à existência da mesma. Neste projecto, optou-se por utilizar informações de temas em que a percentagem de existência de infoboxes é muito grande;
- **Liberdade de esquemas de uma infobox:** uma vez que os utilizadores são livres para criar ou modificar os templates de uma infobox, cada vez que o

utilizador cria ou modifica um tipo de infobox, tem tendência a copiar o modelo da infobox de um artigo semelhante ou alterá-lo de forma a que satisfaça as suas necessidades. Assim, e com o passar do tempo, o esquema da infobox irá ser alterado, pelo que o mapeamento feito para um determinado tipo de infobox seja limitado temporalmente e, no caso de se utilizar a extracção genérica de informações de infoboxes, implica o conhecimento dessas alterações, para reflectir todas as alterações necessárias, para que a utilização dos *placeholders* continue válida;

- **Incorreção das informações da infobox:** tendo em conta que o texto do artigo e a infobox são mantidos separadamente na Wikipédia, e sendo as infoboxes criadas manualmente e não por processos automáticos de extracção de informação do texto, as informações existentes no texto do artigo e na infobox podem ser incoerentes e até contraditórias.

4.2.3 Exemplos de extracções de conhecimento de *infoboxes* elaboradas

Nesta secção irão ser explicados alguns exemplos de extracção de informação de infoboxes sobre alguns temas.

Na Wikipédia, tendo em conta que os textos são estruturados e têm marcações, tais como categoria em que um artigo se encontra, os tópicos relacionados com esse artigo, tais como muitos outros tipos de informações em que o artigo se insere.

Com a atribuição de categorias aos artigos, consegue-se compilar listas de informações de determinadas categorias, o que permite, além de se obter facilmente ligações para outros artigos com a mesma categoria, a sintetização do total de artigos da categoria em questão.

Utilizando este tipo de listas, é muito simples de se obter a totalidade de artigos de um determinado tema, o que permite que se automatizem processos de extracção de informações. Os exemplos de extracção de informação de temas específicos que se seguem, neste caso de países e freguesias de Portugal, foram construídos a partir de tais listas.

Extracção de conhecimento sobre países

Utilizando a “Lista de estados soberanos em 2009”[Wik10a] da Wikipédia, a primeira fase da construção do processo extracção de informações de países consistiu na análise de vários artigos da categoria, de forma a se conseguir obter quais as informações existentes na maioria dos artigos de países, assim como aqueles que poderiam ter mais interesse para utilizar nos diálogos da Etelvina.

Dos vários campos que as infoboxes sobre países possuem, decidiu-se que os campos utilizados na extracção de informações seriam os seguintes: nome nativo

do país, capital, população, língua oficial, gentílico, tipo de governo, líderes do país, moeda, fuso horário, domínio de Internet, indicativo telefónico e fronteiras do país.

Após a escolha dos campos a utilizar na extracção de informação, seguiu-se o levantamento do nome das chaves de cada um dos campos. Com excepção dos líderes do país, todos os outros campos possuem uma chave única e que identifica automaticamente as informações com o seu tipo.

No caso das informações sobre líderes do país, visto que possuem, em vez de um par chave/valor, como os restantes campos, dois pares de chave/valor para definir uma informação, é necessário executar um tratamento específico, tal como citado na secção 4.2.2, dada a existência de vários tipos de governo e, com isso, várias designações para os líderes, assim como variação do número de líderes. Assim, são atribuídos identificadores com as chaves *título_líder* e *nome_líder*, ambos concatenados com um número atribuído sequencialmente, que permite fazer a ligação entre as informações.



De modo a armazenar todas as informações pretendidas na base de dados, criou-se uma tabela com o nome “ConceitosPaíses”, em que os campos correspondem a cada uma das informações da infobox, com uma vez mais a existir uma excepção para os líderes do país. Para evitar a criação de vários campos com os títulos e nomes dos líderes, foi criado um único campo para todos os líderes. Uma vez que as informações obtidas são para a utilização na Etelvina, esse campo é composto por um texto composto pelos vários líderes e títulos. O texto é iniciado com “tem como líder”, sendo concatenado o texto “es” caso exista mais do que um líder. Depois é concatenado o nome dos líderes, assim como o seu título. Ao fim da concatenação do nome e título, é acrescentado, consoante os casos, um ponto final, uma vírgula ou o texto “e”, de forma a se construir uma frase completa. Deste modo, para utilizar no texto a informação dos líderes de um país basta aceder ao campo “Líder” da base de dados e concatenar essa informação. Utilizando o exemplo 4.2, o texto produzido para o campo líder é o seguinte: “tem como líderes Aníbal Cavaco Silva, que é Presidente, e José Sócrates, que é Primeiro-Ministro”.

Após a definição de qual o mapeamento necessário para se conseguir obter informação de uma infobox sobre um país e armazenar correctamente os dados na tabela “ConceitosPaíses”, resta o algoritmo para fazer a pesquisa na Wikipédia de todos os países existentes na lista de países e estados soberanos. Tendo em conta que os países se encontram ordenados alfabeticamente, e que se encontram agrupados pelo primeiro carácter do nome do país, infere-se que existe algum tipo de marca no código-fonte do artigo da Wikipédia para tal ordenação. Assim, verificou-se que cada letra que agrupa um conjunto de países tem a sintaxe “=== letra ===”. No final da lista de países, encontra-se uma nova marcação com o texto “== Outros estados ==”. Assim, tendo marcações para o início e o fim da lista de países, resta efectuar o processamento necessário para extrair informações de todos os artigos da lista.

Cada linha do artigo da lista que corresponde a um país tem a seguinte sintaxe:

```
*'''[[ImagemBandeira]]_[[País]]'''_-_-NomeOficial"-_EtiquetasAnotacoes"
```

Portugal
República Portuguesa





Bandeira Brasão de armas

Lema: *não tem*

Hino nacional: *A Portuguesa*

Gentílico: *português, portuguesa*



Localização de Portugal (em verde)
No continente europeu (em cinzento e verde-claro)
Na União Europeia (em verde-claro)



Capital	Lisboa  38° 42' N 09° 10' O 
Cidade mais populosa	Lisboa <small>(Pop. 2004: 564 657)^[1] Área Metropolitana (Pop. 2006: 2 465 233)</small>
Língua oficial	português ^[2]
Governo	República parlamentar
- Presidente	Aníbal Cavaco Silva
- Primeiro-Ministro	José Sócrates
Formação	1093 d.C.
- Independência ²	1139
- Reconhecida	1143
Entrada na UE	1 de Janeiro de 1986

Figura 4.3: Exemplo de parte da infobox correspondente ao artigo sobre “Portugal”.

Assim, e utilizando uma expressão regular para se obter o nome do país, pesquisa-se a entrada referente do país e depois efectua-se o mapeamento e tratamento das informações.

O tratamento das informações consiste na remoção de marcas que os textos contêm, tais como hiperligações para outros sítios na Internet, notas de rodapé,

referências bibliográficas, entre outros tipos de marcas.

Exemplo 4.3: Extracto de código do artigo "Lista de países e estados soberanos em 2009"[Wik10a].

```

=== O ===
*'''[[Ficheiro:Flag of Oman.svg|22px|border]]
  [[Omã]]''' - Sultanato de Omã <sup>[ONU]</sup>

=== P ===
*'''[[Ficheiro:Flag of the Netherlands.svg|22px|border]]
  [[Países Baixos]]''' - Países Baixos <sup>[ONU]</sup>
*'''[[Ficheiro:Flag of Palau.svg|22px|border]]
  [[Palau]]''' - República do Palau <sup>[ONU]</sup>
*'''[[Ficheiro:Flag of Panama.svg|22px|border]]
  [[Panamá]]''' - República do Panamá <sup>[ONU]</sup>

```

Após todo os tratamentos efectuados, resta armazenar as informações na base de dados. Quando for requerido à Etelvina alguma informação sobre um país, resta à Etelvina fazer uma pesquisa na base de dados e construir o texto de acordo com as informações obtidas, tal como visível na conversa 4.2.3.

Conversa 4.1: Exemplo de conversa com informações de países, obtidos através da extracção de informação de infoboxes da Wikipédia.

Utilizador: Conheces França?

Etelvina: França, ou República Francesa, é um país. O seu nome nativo é République française. Tem 64.473.140 habitantes e a língua oficial é Francês. Os seus habitantes denominam-se de Francês. Possui um governo do tipo unitária semipresidencialismo|semipresidencial, em que tem como líderes Nicolas Sarkozy, que é Presidente, e François Fillon, que é Primeiro-ministro. Euro é a moeda utilizada. Tem o fuso horário CET. Para aceder um site desse país utiliza o domínio .fr e para telefonar o indicativo +33.

Utilizador: e sobre Portugal, sabes algo?

Etelvina: Portugal, ou República Portuguesa, é um país. O seu nome nativo é Portugal. Tem 10 627 250 habitantes e a língua oficial é Português. Os seus habitantes denominam-se de português, portuguesa. Possui um governo do tipo democracia parlamentar, em que tem como líderes José Sócrates, que é Primeiro Ministro, e Aníbal Cavaco Silva, que é Presidente. Euro é a moeda utilizada. Faz fronteira com Espanha. Para aceder um site desse país utiliza o domínio .pt e para telefonar o indicativo +351.

Extracção de conhecimento sobre freguesias de Portugal

À semelhança do processo de extracção de informações de países, utilizou-se um artigo da Wikipédia que possui a lista de de freguesias de Portugal[[Wik10b](#)]. No entanto, existem algumas diferenças entre esta lista e a lista de países e estados soberanos. Enquanto os países se encontravam apenas agrupados alfabeticamente, as freguesias encontram-se agrupadas pelo concelho ao qual pertencem, que por sua vez está agrupado por ordem alfabética. Assim, em vez de se tratar unicamente o nome da freguesia em questão, consegue-se também atribuir o concelho em que está inserida.

No caso das freguesias, o número de dados existente é muito inferior àquele que os países possuem. As informações que se escolheram para obtenção de informação de artigos da Wikipédia relativos a freguesias são o concelho, a área, a população segundo os censos(ano dos censos ao qual o número respeita também é armazenado), a densidade populacional, a data de fundação, o orago (santo festejado na freguesia), o gentílico, o código-postal, o sítio na internet, o email da junta de freguesia e outras informações extra variadas.

Tal como no artigo da lista de países e estados soberanos, o processamento é iniciado quando é encontrada a sintaxe “=== A ===”, mas, no entanto, não possui nenhuma marca para o fim de processamento. Assim, tem de se garantir que as expressões regulares apenas extraem o conteúdo desejado, de forma a não se obter informações erradas sobre as freguesias.

O nome dos concelhos presentes na lista têm a seguinte sintaxe:

```
=== [[Nome do Concelho]] ===
```

Possuindo já o nome do concelho ao qual pertence, resta percorrer todas as entradas do artigo para pesquisar as informações relativas à freguesia. A sintaxe base de cada freguesia é “*[[Nome da Freguesia|Nome do artigo da Wikipédia]]”, existindo no entanto variantes, devido à existência de muitas freguesias portuguesas com nomes iguais, tais como “São Pedro” e “Santa Maria”, que são o nome dezoito e doze freguesias, respectivamente. Para desambiguação de freguesias com o mesmo nome, é adoptado o nome do concelho ao qual pertencem, ficando-se com a sintaxe “*[[Nome da Freguesia(Nome do Concelho)|Nome do artigo]]”. Nestes casos, como a extracção de informações sobre freguesias já continha o nome do concelho ao qual a freguesia pertence, não se extrai o nome do concelho através desta sintaxe. A mesma sintaxe é utilizada quando o nome da freguesia é igual ao nome do concelho, sendo substituído o nome do concelho pelo texto “freguesia”.

Exemplo 4.4: Extracto de código do artigo "Lista de freguesias de Portugal"[[Wik10b](#)].

```
=== [[Trofa]] ===
* [[Alvarelhos (Trofa)|Alvarelhos]]
* [[Santiago de Bougado|Bougado (Santiago)]]
* [[São Martinho de Bougado|Bougado (São Martinho)]]
```

```

*[[São Mamede de Coronado|Coronado (São Mamede)]]
*[[São Romão do Coronado|Coronado (São Romão)]]
*[[Covelas (Trofa)|Covelas]]
*[[Guidões]]
*[[Muro]]

```

```

== V ==

```

```

=== [[Vagos]] ===
*[[Calvão (Vagos)|Calvão]]
*[[Covão do Lobo]]
*[[Fonte de Angeão]]
*[[Gafanha da Boa Hora]]
*[[Ouça]]
*[[Ponte de Vagos]]
*[[Santa Catarina (Vagos)|Santa Catarina]]
*[[Santo André de Vagos]]
*[[Santo António de Vagos]]
*[[Sosa]]
*[[Vagos (freguesia)|Vagos]]

```

Como não existem campos dinâmicos, com excepção do campo extra, composto unicamente por texto livre, o mapeamento das informações é directo para a tabela da base de dados, obtendo-se assim bastante informação sobre 4259 freguesias portuguesas, que é o número actual de freguesias no país.

4.2.4 Extracção de conhecimento de páginas genéricas da Wikipédia

Quando uma artigo da Wikipédia não possui textos reconhecíveis pelas técnicas de extracção de informação referidas nas secções anteriores, tais como as marcas das infoboxes e textos estruturados, torna-se impossível extrair informação estruturada do artigo.

Ao se limitar a extracção de conhecimento a temas em que existam infoboxes que sejam reconhecíveis pela Etlvina, a quantidade de informação que possui é muito baixa, pois implica que os artigos obrigatoriamente possuam as tais infoboxes de um determinado tema, que tenha sido definido quais os campos a extrair-se e qual a metodologia de extracção utilizada para a extracção e utilização de informações do artigo. Isso implica que a base de conhecimento da Etlvina será resumida a um número de temas limitado, limite esse imposto pela quantidade de temas que o utilizador tenha definido técnicas de extracção, mas com um elevado nível de estruturação, o que permite utilizar as várias informações individualmente ou associadas às restantes informações obtidas através do artigo da Wikipédia do conceito em questão.

De forma a que a extracção de informação não seja limitada unicamente a informações estruturadas, decidiu-se implementar uma metodologia de extracção de conhecimento através de artigos genéricos, isto é, sem qualquer tipo de marcação, nem de infoboxes já reconhecidas.

Para a extracção do conhecimento de páginas genéricas, foi necessário saber como extrair informações que fossem relevantes de cada artigo, sem extrair a totalidade do texto do artigo. Assim, e após uma análise detalhada a vários artigos, definiu-se que, tendo em conta a estrutura comum dos artigos da Wikipédia, o primeiro parágrafo é aquele que contém a definição, de um modo geral, do termo pesquisado. Com isto, quando o módulo da Etlvina de tratamento de informações da Wikipédia recebe um texto, o módulo pesquisa, utilizando um algoritmo semelhante ao descrito na secção 4.2.2.

A Etlvina, ao receber uma mensagem de entrada do utilizador e que ainda não foi respondida por nenhum módulo, divide a mensagem pelas várias palavras que a compõe. Após ter todas as palavras separadas, começa a pesquisar, tanto na tabela de Conceitos da base de dados da Etlvina, como na Wikipédia, em tempo de execução, se possui alguma informação para a primeira palavra da mensagem de entrada. Caso exista alguma informação associada àquela palavra, quer na Wikipédia, quer na base de dados, é adicionada a uma estrutura as seguintes informações:

- Termo pesquisada;
- Informação obtida sobre o termo em questão;
- Percentagem da relevância do termo pesquisado na mensagem de entrada.

A relevância do termo é utilizado para que o resultado devolvido seja a expressão mais comprida pesquisada, e, em caso de haver expressões com comprimento igual, será sorteada entre essas ocorrências.

Esta percentagem apenas surgiu numa fase final do projecto, pois não se tinha analisado quais os problemas de se sortear apenas um termo entre as várias possibilidades existentes na frase do utilizador. De forma a ilustrar um dos casos que incentivou a criação de um critério de devolução de respostas mais acertivo, segue-se um exemplo.

Ao utilizar a Etlvina com o módulo de extracção de informação da Wikipédia, denotava-se que a quantidade de informação que a Etlvina possuía, através de pesquisas na Wikipédia, era bastante grande. No entanto apenas devolvia a ocorrência desejada em dois momentos:

1. Quando a entrada é muito específica e faz com que apenas exista uma ocorrência do termo, sem que exista informação de nenhuma das palavras pertencentes ao termo pesquisado;
2. Quando, por sorte, perante um determinado número de ocorrências para o termo e todas as palavras que fazem parte do termo, a mensagem devolvida

pela Etelvina da lista de respostas possíveis coincida precisamente com a referência que o utilizador desejava obter informação.

Para se exemplificar a aleatoriedade da escolha da resposta entre as respostas possíveis, seguem-se pequenos blocos de conversa de um utilizador com a Etelvina.

Conversa 4.2: Exemplo de uma resposta da Etelvina em que o resultado não é o pretendido.

Utilizador: conheces a música lucy in the sky with diamonds dos the beatles?

Etelvina: 'THE' é um sistema operacional desenvolvido por um time liderado por Edsger Dijkstra, descrito em monografia entre 1965 e 1966 e publicado em 1968. Dijkstra nunca nomeou o sistema; "THE" é simplesmente a abreviação de "Technische Hogeschool Eindhoven", o nome em neerlandês da Universidade de Tecnologia de Eindhoven, nos Países Baixos. O sistema THE é basicamente um sistema batch que suporta multitarefa; não foi desenvolvido para se multiusuário.

Nesta parte da conversa, o utilizador pretende informações sobre a música "Lucy in The Sky With Diamonds", da banda "The Beatles". As respostas possíveis que a Etelvina possuía eram:

- "Lucy in the Sky with Diamonds" é o título de uma canção composta e gravada pelos Beatles em 1967, e faz parte do oitavo álbum da banda, Sgt. Pepper's Lonely Hearts Club Band.
- "The Beatles" foi uma banda de rock de Liverpool, Inglaterra, com suas raízes no final da década de 1950 e formada na década de 1960, Suas vestimentas, seus cortes de cabelo e sua crescente consciência social exerceram forte influência sobre a juventude da época; criou-se o termo "beatlemania" para designar o intenso fanatismo pela banda demonstrado pelos jovens. Após a banda se separar em 1970, os quatro membros iniciaram carreiras solo de sucesso.
- 'THE' é um sistema operacional desenvolvido por um time liderado por Edsger Dijkstra, descrito em monografia entre 1965 e 1966 e publicado em 1968. Dijkstra nunca nomeou o sistema; "THE" é simplesmente a abreviação de "Technische Hogeschool Eindhoven", o nome em neerlandês da Universidade de Tecnologia de Eindhoven, nos Países Baixos. O sistema THE é basicamente um sistema batch que suporta multitarefa; não foi desenvolvido para se multiusuário.

Perante as respostas possíveis, a Etelvina, apesar de possuir duas respostas que poderiam satisfazer o utilizador, devolveu uma resposta que em nada se encaixava com o contexto da conversa.

Tentando novamente obter alguma informação sobre a banda, tentou-se refinar a entrada. Mesmo assim o resultado continuou sem ser o pretendido, uma vez que a Etelvina encontrou uma ocorrência para o termo falar e, na hora da resposta, ao escolher aleatoriamente se devolvia a informação sobre os “The Beatles” ou o termo “falar”, a escolha/sorte fez com que a resposta devolvida fosse novamente algo fora de contexto.

Conversa 4.3: Outro exemplo de uma resposta da Etelvina em que o resultado não é o pretendido.

Utilizador: estava a falar dos the beatles...

Etelvina: 'Diálogo' é uma conversação estabelecida entre duas ou mais pessoas. Como tal, é a principal forma de criação do teatro (embora o monólogo tenha neste gênero de arte também o seu lugar de destaque). Para existir diálogo tem de existir um locutor e um interlocutor

Por fim, e de forma a se conseguir concluir se a Etelvina possuía informação sobre a banda, decidiu-se apenas enviar na mensagem o termo “the beatles”. Só assim se conseguia obter, com certeza, o resultado pretendido, apesar de a Etelvina possuir como uma das respostas possíveis desde a primeira questão aquela que seria considerada correcta, por se encontrar no contexto da conversa.

Conversa 4.4: Exemplo de um resultado obtido conforme o esperado, dado a especificidade do termo na mensagem de entrada.

Utilizador: the beatles

Etelvina: 'The Beatles' foi uma banda de rock de Liverpool, Inglaterra, com suas raízes no final da década de 1950 e formada na década de 1960, Suas vestimentas, seus cortes de cabelo e sua crescente consciência social exerceram forte influência sobre a juventude da época; criou-se o termo "beatlemania" para designar o intenso fanatismo pela banda demonstrado pelos jovens. Após a banda se separar em 1970, os quatro membros iniciaram carreiras solo de sucesso.

Após a atribuição da percentagem de relevância na mensagem de entrada de cada um dos termos pesquisados, os resultados obtidos para as mesmas mensagens foram os presentes na conversa 4.5.

Conversa 4.5: Exemplo de respostas devolvidas pela Etelvina, após a implementação da percentagem de relevância.

Utilizador: conheces a música lucy in the sky with diamonds dos the beatles?

Etelvina: "'Lucy in the Sky with Diamonds'"é o título de uma canção composta e gravada pelos Beatles em 1967, e faz parte do oitavo álbum da banda, Sgt. Pepper's Lonely Hearts Club Band.

Utilizador: estava a falar dos the beatles...

Etelvina: 'The Beatles' foi uma banda de rock britânica, formada em Liverpool em 1960 e um dos atos mais comercialmente bem-sucedidos e aclamados da história da música popular. A partir de 1962, o grupo era formado por John Lennon (guitarra rítmica e vocal), Paul McCartney (baixo e vocal), George Harrison (guitarra solo e vocal) e Ringo Starr (bateria e vocal). Enraizada do skiffle e do rock and roll da década de 1950, a banda veio mais tarde a assumir diversos gêneros que vão do folk rock ao rock psicodélico, muitas vezes incorporando elementos da música clássica e outros em formas inovadoras e criativas. Sua crescente popularidade, que a imprensa britânica chamava de "Beatlemania", fizeram com que eles crescessem em sofisticação. Os Beatles vieram a ser percebidos como a encarnação de ideais progressistas e sua influência se estendeu até as revoluções sociais e culturais da década de 1960.

Utilizador: the beatles

Etelvina: 'The Beatles' foi uma banda de rock britânica...

Como é possível verificar no exemplo anterior, a adição de um valor de comparação que consiga definir qual a melhor resposta melhora bastante os resultados. No entanto, para o sistema ser ainda mais acertivo, seria necessário atribuir diferentes valores a cada palavra, pois palavras como "a", "o", "os", "e", entre outras, são bastante frequentes na língua portuguesa e, no actual método, têm o mesmo peso que as restantes.

Após o tratamento para a primeira palavra estar concluído, é executado um ciclo de pesquisas, em que o termo corresponde inicialmente à primeira e à segunda palavra, sendo finalizado o ciclo com a pesquisas com os termos a serem compostos pela primeira palavra até às sete seguintes (o motivo para a escolha de oito palavras encontra-se na secção 4.2.2). O mesmo procedimento é executado para a segunda palavra e assim consecutivamente, até serem pesquisados todos os termos e as sete palavras seguintes da mensagem de entrada.

Após a pesquisa de todos os termos estar terminada, a Etelvina, com base na percentagem de relevância, testa qual a(s) resposta(s) com uma percentagem maior. Se apenas existir uma resposta, a mensagem enviada ao utilizador é igual a essa resposta. Caso existam várias, será sorteado do conjunto obtido.

Conversa 4.6: Exemplo de pesquisas na Wikipédia, em páginas genéricas, através da Etelvina.

Utilizador: sapo

Etelvina: O nome 'sapo' é uma designação genérica de anfíbios da ordem Anura, e em especial da família 'Bufonidae'. No entanto, não sendo uma designação científica, aplica-se também a outras famílias. Por exemplo, o sapo-parteiro pertence à família Discoglossidae, à qual pertencem também as rãs-pintadas.

Utilizador: Bracara Augusta

Etelvina: 'Bracara Augusta', o nome romano da actual cidade de Braga, no norte de Portugal, foi construída no lugar de um povoado anterior de origem celta. A cidade romana foi fundada pelo imperador César Augusto cerca de 16 aC, após a pacificação definitiva da região. Durante o período dos Flávios, Bracara Augusta recebeu o estatuto municipal e foi elevada a sede do conventus, tendo tido funções administrativas sobre uma extensa região. A partir da reforma de Diocleciano passou a ser a capital da recente província da Galécia. No século V a cidade foi tomada pelos invasores suevos, que a escolheram como capital do seu reino.

Utilizador: Universidade do Minho

Etelvina: A 'Universidade do Minho' ('UM') foi fundada em Braga em 1973 e integrou-se no chamado grupo das "Novas Universidades" que vieram alterar o panorama do ensino superior em Portugal. Iniciou as suas actividades académicas em 1975/76. A universidade é governada por uma reitoria, composta por um reitor e um concelho geral, e cinco unidades internas que agrupam por áreas de intervenção as várias entidades internas.

Apesar de a metodologia escolhida para páginas genéricas conseguir ter bons resultados para a pesquisa de artigos através da Etelvina, continuam a existir limitações, tal como existiam na obtenção de informações através de infoboxes. Assim sendo, segue uma enumeração dos problemas obtidos com esta técnica utilizada:

1. **As informações continuam a não ser fidedignas:** esta limitação está inerente à Wikipédia, e apenas com a boa fé dos utilizadores e de controlo de entradas é que é possível não existir;
2. **Nem sempre o primeiro parágrafo obtido tem informação:** por vezes, o primeiro parágrafo de uma entrada na Wikipédia não possui o resumo da informação sobre a entrada, como, por exemplo, temas em que o primeiro parágrafo é uma legenda de uma imagem;
3. **Redireccionamentos:** quando se fazem pesquisas na Wikipédia por apenas algumas partes do termo do artigo que se deseja obter informações, existem várias páginas de redireccionamento. Enquanto que navegando na Wikipédia através de um *browser* esse redireccionamento ocorre automaticamente, sendo por vezes transparente ao utilizador, ao se pesquisar textualmente através de

APIs, o resultado obtido apenas contém o texto “*#REDIRECT termo*” ou “*#REDIRECIONAMENTO termo*”, em que o termo corresponde ao nome do artigo que a Wikipédia redirecciona;

4. **Quando existem entradas ambíguas, não se obtém nenhuma informação útil no primeiro parágrafo:** na Wikipédia, quando existem entradas ambíguas, torna-se complicado de conseguir obter alguma informação. Quando se utiliza a Wikipédia num *browser*, é aberta uma página de desambiguação, em que aparecem ligações para todas as entradas que possuam aquele(s) termo(s). Com a técnica de extracção do primeiro parágrafo obtém-se apenas mensagens do tipo “*termo pesquisado pode referir-se a:*”.

Perante tais problemas, tentou-se ultrapassar as limitações existentes, de forma a se poder obter um sistema com uma percentagem de respostas aceitável. Assim, e perante os problemas supracitados, foram criados mecanismos que conseguissem evitar esses problemas, com excepção do primeiro ponto, pois evitar tal problema é impossível, dado ser um problema que a Wikipédia sempre possui e continuará a possuir, uma vez que a enciclopédia é aberta para os utilizadores (só com a colaboração dos utilizadores, que criaram milhares de artigos com informações fidedignas, é que conseguiu crescer).

Em relação ao segundo ponto, conseguiu-se resolver parcialmente o problema, ignorando todas as imagens, infoboxes e outro tipo de objecto com marcações especiais antes do primeiro parágrafo. Assim, consegue-se obter o texto do parágrafo, no entanto nada impede que o parágrafo não possua informação útil e relevante, sendo que se torna impossível de resolver esse problema programaticamente em tempo útil.

Para a resolução do terceiro ponto, foram criados testes específicos para os textos dos artigos em questão. Uma vez que resultado da pesquisa do artigo possui uma marca para informar que se trata de um redireccionamento (as marcas “*#REDIRECT*” e “*#REDIRECIONAMENTO*”,) e de seguida o termo para o qual redirecciona, foi criado um teste para verificar se o texto do artigo possui uma marca de redireccionamento. Caso essa marca exista, extrai-se o termo para o qual deve ser feito o redireccionamento, e pesquisa-se esse mesmo termo obtido na Wikipédia. Deste modo, todos os artigos de redireccionamento são utilizados como se estivesse a navegar na Wikipédia através de um browser, resolvendo este problema.

Na tentativa de resolução do quarto ponto, inicialmente foram criados testes para algumas frases de desambiguação, testando-se se a mensagem possuía o texto composto pelo termo, seguido das palavras “pode-se referir a”. Apesar de responder às necessidades dos primeiros termos pesquisados que tivessem uma página de desambiguação, uma pesquisa mais intensa sobre este tipo de páginas levou à descoberta de uma página na Wikipédia com todas as páginas de desambiguação existentes. Analisando essa página de desambiguações da Wikipédia[Wik10c], verificou-se que existiam, à data de escrita deste documento, 15895 páginas de desambiguação, em que cada página possui pelo menos a desambiguação de dois termos.

Dado o número de páginas que a Etelvina não conseguiria tratar, foi feita uma análise mais profunda a várias páginas de desambiguação para se detectar de um padrão que permita que a Etelvina devolva ao utilizador, como resposta a um termo em que o resultado obtido da pesquisa na Wikipédia corresponda a uma página de desambiguação.

As páginas de desambiguação da Wikipédia iniciam-se todas com o texto “*{{desambiguação}}*”, possuindo variados formatos de apresentação da lista de resultados possível, sendo por vezes agrupadas por temas, tais como, caso seja uma desambiguação geográfica, o país onde se encontra a localidade ambígua, ou, na maioria das vezes, simplesmente uma lista com os resultados obtidos e uma pequena definição, de forma a que o utilizador consiga encontrar informação sobre o artigo que pesquisa.

Em cada linha que represente um resultado de uma desambiguação, foi detectado que existe uma marca no início de cada, marca essa definida pelo carácter '*'. Assim, estando numa página de desambiguação, é analisado todo o documento, verificando quais as linhas que sejam iniciadas pelo carácter '*'. Quando essa marca existir, acrescenta-se um texto com o novo artigo, assim como uma pequena definição do termo em questão. Após o tratamento de todas as linhas da página de desambiguação, é devolvido ao utilizador um texto a alertar que o termo que foi pesquisado é ambíguo e que se pode referir a vários artigos, utilizando-se os textos das várias linhas iniciadas por '*'.

Conversa 4.7: Exemplo de uma resposta da Etelvina com a listagem de resultados de artigos ambíguos.

Utilizador: Boavista

Etelvina: O termo Boavista pode referir-se a:

- 1: Boavista (pesquisar por 'Boavista (Moçambique)'), uma localidade da província de Manica.
- 2: Avenida da Boavista, na cidade do Porto
- 3: Boavista (pesquisar por 'Boavista (Lousada)'), lugar da freguesia do Torno, em Lousada
- 4: Boavista (pesquisar por 'Boavista (Porto)'), designação genérica de uma parte da zona ocidental da cidade do Porto, em Portugal.
- 5: FC Boavista, clube multiesportes da cidade da Praia, na Ilha de Santiago.
- 6: Boavista Futebol Clube, um clube de futebol da freguesia de Ramalde, no concelho de Porto.
- 7: Boavista Sport Clube, clube desportivo com sede em Santa Cruz das Flores, ilha das Flores, arquipélago dos Açores.
- 8: Boavista Sport Club, um clube de futebol da cidade de Saquarema, no estado do Rio de Janeiro.
- 9: Boa Vista

Utilizador: A1

Etelvina: O termo A1 pode referir-se a:

- 1: A1 (tamanho de papel), tamanho de papel.
- 2: A1 (leite), proteína do leite.

- 3: A1 (míssil), míssil alemão
- 4: A1 (canal de tv), canal de tv da Macedónia
- 5: A1 (autoestrada), autoestrada portuguesa que liga o Porto a Lisboa.
- 6: A1 Grand Prix, prova de automobilismo de monolugar.

Através deste mecanismo de desambiguação de resultados, consegue-se obter informação sobre milhares de termos, que, analisando apenas o primeiro parágrafo, não seria possível.

Capítulo 5

Conclusões

Ao longo deste documento foram apresentadas as etapas da construção do motor para a criação de Elizas, tendo sempre presente que os principais objectivos eram a possibilidade do motor ser configurável, assim como ser facilmente expansível.

Apresentaram-se algumas das possíveis estratégias, entre as quais as adoptadas para o desenvolvimento da Etelvina, para a criação de uma Eliza com a capacidade de se tornar numa ferramenta útil para possuir na lista de contactos de um cliente de mensagens instantâneas, e também com a capacidade de oferecer momentos de entretenimento aos utilizadores.

A estrutura central adoptada, designada de motor, responde ao requisitos de expansibilidade, pois permite a ligação de novos módulos de funcionalidades, e também ao de configurabilidade, pois é através de linguagens de configuração que se define quais os módulos que se pretende utilizar, quais as definições base necessárias para a ligação a um cliente de mensagens instantâneas e também a configuração dos vários módulos, tornando relativamente simples a adaptação da Etelvina para outras línguas, por exemplo.

A necessidade de se poder alterar facilmente os comportamentos da personagem levou à criação de uma linguagem de domínio específico. Esta linguagem foi pensada de forma a ser o mais simples possível. Tendo em conta o tipo de acções que o programa tem de executar, assumiu-se que a linguagem devia ter capacidade para reconhecer expressões de entrada, executar cálculos ou transformações, e, conseqüentemente, devolver uma resposta. A utilização desta linguagem permitiu que os módulos que compõe a Etelvina se tenham tornado versáteis e simples de implementar novas regras.

Utilizando como comparação o AIML, que é a linguagem que mais se assemelha à linguagem criada, pode-se concluir que esta última é bastante mais simples, além de ser mais fácil a adição de comportamento dinâmico, devido aos blocos de código existentes nas regras. No entanto, e apesar de a Etelvina possuir uma maior capacidade de armazenamento das últimas mensagens recebidas, o AIML, possuindo apenas a última, consegue utilizá-la de forma mais simples para a manutenção do

tema da conversa.

A falta de configurações simples para a manutenção do tema da conversa é, possivelmente, uma das maiores lacunas da Etelvina, uma vez que é necessário utilizar o bloco de código de uma regra para se conseguir consultar o histórico. Com esta limitação, torna-se complicado conseguir gerar um modo de entrevista, em que as perguntas sejam encadeadas, do género de perguntas como se o utilizador gosta de cinema e, em caso afirmativo, qual o seu filme favorito.

Apesar de em AIML este tipo de configuração ser simples, implica a atribuição de um padrão a todas as regras, o que torna mais morosa a tarefa de configuração de regras mesmo que não utilizem a última frase.

A construção de uma base de conhecimento foi a tarefa que envolveu mais tempo de preparação, desde a análise até à tomada de decisões, dadas as inúmeras possibilidades existentes para conseguir executar a extracção de informação. A investigação sobre sistemas de respostas a perguntas foi demorada, dada a complexidade desta área de investigação. Foi através dessa análise que se conseguiu obter dados muito interessantes sobre a classificação de perguntas para extracção de informação, no entanto este tipo de abordagem implicava a criação de um novo projecto, que teria obrigatoriamente de ser concretizado por uma equipa com várias pessoas e com um espaço temporal mais alargado.

As técnicas adoptadas para extracção de conhecimento da Wikipédia permitiram a construção de uma base de dados de termos interessante, do ponto de vista de utilização em conversas com a Etelvina, pois consegue-se obter informações simples e actuais, e utilizar os blocos dos artigos da Wikipédia para devolver ao utilizador uma resposta com o conteúdo pretendido por este, na maior parte dos casos.

A criação dos vários módulos desenvolvidos no decorrer do projecto, utilizando toda a estrutura criada, para a criação da Etelvina (personagem), serviu como uma prova de conceito da expansibilidade e configurabilidade do sistema.

Mesmo com as limitações existentes, conclui-se que a Etelvina consegue ser uma base para a implementação de Elizias, que, com as configurações adequadas para cada utilizador, torna possível, através de um cliente de mensagens instantâneas, ter acesso a um conjunto de funcionalidades úteis para o dia-a-dia.

5.1 Trabalho Futuro

O estado actual do projecto permite que se tenha a ambição de se continuar a desenvolver a Etelvina, de forma a aproximá-la o mais possível dos objectivos propostos e da possibilidade de ser uma estrutura de referência para a criação de Elizias. Desta forma, ficam em agenda os seguintes trabalhos:

- Criação de uma ferramenta que valide a sintaxe de ficheiros de configuração, de forma a que o utilizador tenha a garantia de que vai obter os comportamentos desejados e por ele configurados;

- Contextualização do tema da conversa, de forma a se conseguir obter respostas que se enquadrem com o tema do diálogo, criando a capacidade de atribuir um contexto à conversa;
- Reacção de eventos do utilizador, tais como se o utilizador está a escrever ou parou de escrever, de forma a poder esperar para devolver uma resposta, ou mesmo questionar o utilizador porque está a escrever há tanto tempo ou porque parou de escrever;
- A utilização de conversas multi-utilizador, que permita que a Etelvina participe em salas de conversação e saiba a quem se dirige uma determinada mensagem, de forma a responder não só para um utilizador, mas para todos, caso seja necessária a sua participação na conversa;
- O suporte a um maior número de linguagens de programação nos blocos de código, de forma a que se consiga atrair uma comunidade de utilizadores maior, além de um número de funcionalidades superior;
- A extracção de informação de várias fontes e com diferentes técnicas, de forma a poder obter resultados mais exactos, além de um maior número de termos conhecidos.

Referências Bibliográficas

- [ACF⁺07] Carlos Amaral, Adán Cassan, Helena Figueira, André F. T. Martins, Afonso Mendes, Pedro Mendes, Cláudia Pinto, and Daniel Vidal. Priberam's question answering system in qa@clef 2007. In Carol Peters, Valentin Jijkoun, Thomas Mandl, Henning Müller, Douglas W. Oard, Anselmo Peñas, Vivien Petras, and Diana Santos, editors, *CLEF*, volume 5152 of *Lecture Notes in Computer Science*, pages 364–371. Springer, 2007.
- [AFM⁺05] Carlos Amaral, Helena Figueira, André F. T. Martins, Afonso Mendes, Pedro Mendes, and Cláudia Pinto. Priberam's question answering system for portuguese. In *CLEF*, pages 410–419, 2005.
- [Alm10] Alberto Almeida, José João e Simões. Text::rewriterules. <http://search.cpan.org/dist/Text-RewriteRules/>, 2010.
- [Bod08] Margaret Boden. *Mind as Machine: A History of Cognitive Science*. Oxford University Press, Inc., New York, NY, USA, 2008.
- [Car10] Rollo Carpenter. Jabberwacky. <http://www.jabberwacky.com/>, 2010.
- [CC10] Coca-Cola. Coca-cola – contact us. <http://questions.coca-cola.com/>, 2010.
- [Col75] Kenneth Mark Colby. *Artificial Paranoia: A Computer Simulation of Paranoid Processes*. Elsevier Science Inc., New York, NY, USA, 1975.
- [Eat10] Ryan Eatmon. Net-xmpp library. <http://search.cpan.org/dist/Net-XMPP/>, 2010.
- [Fer06] Joaquim Quinteiro Ferreira, Leandro Padilha; Uchôa. Desenvolvimento de um chatbot para auxiliar o ensino de espanhol como língua estrangeira. In *Bazar: Software e Conhecimento Livres*, 2006.
- [GTa10] GTalk. About google talk extensions. http://code.google.com/intl/es/apis/talk/jep_extensions/extensions.html, 2010.

- [Her01] Ulf Hermjakob. Parsing and question classification for question answering. In *Proceedings of the Workshop on Open-Domain Question Answering at ACL-2001*, 2001.
- [IKE10] IKEA. Anna. <http://www.ikea.com/pt/>, 2010.
- [Inc10] Zabaware Inc. Ultra hal. <http://www.zabaware.com/home.html>, 2010.
- [Jab09] Jabber. Jabber software foundation. <http://www.jabber.org>, 2009.
- [JS97] W. Lewis Johnson and Erin Shaw. Using agents to overcome deficiencies in web-based courseware. In *Proceedings of the workshop "Intelligent Educational Systems on the World Wide Web" at the 8th World Conference of the AIED Society, IV-2*, pages 18–22, 1997.
- [Kol93] Janet Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publishers, Inc, 1993.
- [Lav10] Simon James Laven. The simon leave page. <http://www.simonlaven.com>, 2010.
- [Lev10] Dr. David Levy. Do-much-more. <http://www.worldsbestchatbot.com/>, 2010.
- [Luc10] LucasArts. Lucas arts entertainment company | "yoda's help desk" knowledge base. <http://www.lucasarts.com/support/>, 2010.
- [Mau94] Michael L. Mauldin. Chatterbots, tinymuds, and the turing test: entering the loebner prize competition. In *AAAI '94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, pages 16–21, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.
- [PAI10] PAI. Buscas. <http://www.buscas.pt/>, 2010.
- [Rap76] Bertram Raphael. *Thinking Computer: Mind inside Matter*. W. H. Freeman & Co., New York, NY, USA, 1976.
- [RJ97] Jeff Rickel and W. Lewis Johnson. Integrating pedagogical capabilities in a virtual environment agent. In *AGENTS '97: Proceedings of the first international conference on Autonomous agents*, pages 30–38, New York, NY, USA, 1997. ACM.
- [Rob10] Fred Roberts. Elbot - artificial intelligence. <http://www.elbot.com/>, 2010.

- [Rog61] Carl R. Rogers. *On becoming a person : a theraphist's view of psychotherapy*. Houghton Mifflin, Boston :, 1961.
- [RWTB10] Thomas Ringate, Richard S. Wallace, Anthony Taylor, and Jon Baer. Aimpl reference manual. <http://www.alicebot.org/documentation/aiml-reference.html>, 2010.
- [SL00] Rohini Srihari and Wei Li. A question answering system supported by information extraction. In *Proceedings of the sixth conference on Applied natural language processing*, pages 166–172, Morristown, NJ, USA, 2000. Association for Computational Linguistics.
- [Teu04] Christof Teuscher. *Alan Turing: Life and Legacy of a Great Thinker*. SpringerVerlag, 2004.
- [Tur50] Alan M. Turing. Computing machinery and intelligence. *Mind*, LIX:433–460, 1950.
- [Wal00] Richard S. Wallace. Don't read me - a.l.i.c.e. and aiml documentation. <http://www.alicebot.org/documentation/dont.html>, 2000.
- [Wei66] Joseph Weizenbaum. Eliza: A computer program for the study of natural language communication between man and machine. *Communciations of the ACM*, 9(1), 1966.
- [Wik10a] Wikipedia. Lista de estados soberanos em 2009. http://pt.wikipedia.org/wiki/Anexo:Lista_de_estados_soberanos_em_2009, 2010.
- [Wik10b] Wikipedia. Lista de freguesias de Portugal. http://pt.wikipedia.org/wiki/Anexo:Lista_de_freguesias_de_Portugal, 2010.
- [Wik10c] Wikipedia. Sobre as desambiguações da Wikipedia. <http://pt.wikipedia.org/wiki/Categoria:Desambiguação>, 2010.

Anexos

A. Conversa entre a *Eliza* e o *Parry*, na ICCC de 1972.

Eliza: Good Evening.
Tell me your problems.
Please terminate input with a period or a question mark.

Parry: People get on my nerves sometimes.

Eliza: I am not sure I understand you fully.

Parry: You should pay more attention.

Eliza: Suppose you should pay more attention.

Parry: You're entitled to your own opinion.

Eliza: What makes you think I am entitled to my own opinion?

Parry: It bothers me just to be around people in general.

Eliza: Please go on.

Parry: I went to the track at Bay Meadows a while back.

Eliza: What does that suggest to you?

Parry: I went to the races.

Eliza: What else would you like to discuss?

Parry: Horseracing interests me.

Eliza: Why do you say that just now?

Parry: There's a lot of crooked gambling going on there.

Eliza: I am not sure I understand you fully.

Parry: It's clear to me if not to you.

Eliza: Do you think its likely that not to I?

Parry: I went to the races.

Eliza: Please go on.

Parry: I gamble on horses.

Eliza: What does that suggest to you?

Parry: I went to the races.

Eliza: What else would you like to discuss?

Parry: Gambling has been nothing but trouble to me.

Eliza: Why do you say that just now?

Parry: I've had experience with crooks gambling at the track.

Eliza: I am not sure I understand you fully.

Parry: I've already told you enough about that.

Eliza: Oh? I enough about that?

Parry: I went to the races.

Eliza: Please go on.

Parry: Do you know anything about bookies?
Eliza: What makes you think I know anything about bookies?
Parry: A bookie didn't pay me off once.
Eliza: What does that suggest to you?
Parry: I beat up the bookie who did not pay off, and I thought he might use his friends in the underworld to get even with me.
Eliza: What else would you like to discuss?
Parry: What do you know about racketeers?
Eliza: Why do you ask?
Parry: What experience have you had gambling?
Eliza: Why does that question interest you?
Parry: I like horseraces.
Eliza: What else do you like?
Parry: I know the mob controls the big rackets.
Eliza: Who are you really talking about?
Parry: Have you ever really thought about the MAFIA?
Eliza: Suppose you ever really thought about the MAFIA.
Parry: Let's talk about something else.
Eliza: I am not sure I understand you fully.
Parry: You keep asking me the same things, why not change the topic?
Eliza: In what way?
Parry: What do you mean?
Eliza: Why are such questions on your mind?
Parry: I thought I told you, people get on my nerves.
Eliza: Your nerves?
Parry: Well, I don't know.
Eliza: Don't you really know?
Parry: I would say so.
Eliza: Please go on.
Parry: I would rather not discuss that anymore.
Eliza: Would you like to discuss your nerves?
Parry: I guess so.
Eliza: What does that suggest to you?
Parry: What are you getting at?
Eliza: What answer would please you most?
Parry: It's not a matter of like or dislike.
Eliza: What else would you like to discuss?
Parry: I am getting tired of answering the same questions over and over.
Eliza: What resemblance do you see?
Parry: Why do you need to know that?
Eliza: What do you think?
Parry: I don't understand your motives.
Eliza: Why don't you understand my motives?
Parry: Shouldn't I be?
Eliza: Why do you say that just now?
Parry: That's just the way it is.
Eliza: Suppose that's just the way it were not.
Parry: So?
Eliza: I am not sure I understand you fully.
Parry: You harp on the same questions too much.
Eliza: What does that similarity suggest to you?

Parry: I don't get you.
Eliza: Do you wish that you did get I?
Parry: It's about time you showed some interest in my feelings.
Eliza: Oh? I showed some interest in your feelings?
Parry: I will help you if I trust you.
Eliza: Do you wish that you trust I?
Parry: Maybe yes, maybe no.
Eliza: You don't seem quite certain.
...Excuse me for just a minute.
Sorry to have interrupt you please continue...
Parry: Really?
Eliza: Please go on.
Parry: In my opinion you have a one-track mind.
Eliza: What makes you think I have a one-track mind?
Parry: I used to go to the track every week.
Eliza: What does that suggest to you?
Parry: I avoid racetracks now because the races are fixed.
Eliza: Is that the real reason?
Parry: I'd rather not discuss it.
Eliza: What else would you like to discuss?
Parry: That's something I don't want to go into.
Eliza: Does that trouble you?
Parry: I went to the races.
Eliza: Why do you say that just now.
Parry: It's none of your business.
Eliza: Why are you concerned over my business?
Parry: I don't confide in strangers.
Eliza: Don't you really confide in strangers?
Parry: Let's change the subject.
Eliza: Perhaps that concerns your feelings.
Parry: I went to the races.
Eliza: I am not sure I understand you fully.
Parry: What's wrong with you that you keep on bugging me about this?
Eliza: Who are you really talking about?
Parry: The way you repeat yourself is getting ridiculous.
Eliza: Suppose you repeat myself is getting ridiculous.
Parry: There are some tricky operators down there.
Eliza: Are there really some tricky operators down there?
Parry: The guys involved in the horseracing racket.
Eliza: Please go on.
Parry: You are a real nag.
Eliza: Does it please you to believe I am a real nag?
Parry: I have had enough of this.
Eliza: What does that suggest to you?
Parry: Goodbye.

B. Letra da música “*Etelvina*”

Etelvina com seis meses já se tinha de pé
Foi deixada num cinema depois da matinée
Com um recado na lapela que dizia assim
Quem tomar conta de mim,
quem tomar conta de mim
Saiba que fui vacinada,
Saiba que sou malcriada.

Etelvina com dezasseis anos já conhecia
Todos os reformatórios da terra onde vivia
Entregaram-na a uma velha que ralhava assim
“Ai menina sem juízo
nem mereces um sorriso
Vais acabar num bueiro
sem futuro nem dinheiro.”

“Eu durmo sozinha à noite
Vou dormir à beira rio à noite, à noite
Acocorada com o frio à noite, à noite”

Etelvina era da rua como outros são do campo
Sua cama era um caixote sem paredes nem tampo
Sua janela uma ponte que dizia assim:
“Dentro das minhas cidades
já não sei quem é ladrão
Se um que anda fora de grades
se outro que está na prisão”

Etelvina só gostava era de andar pela cidade
A semear desacatos e a colher tempestades
A meter-se cós ricaços, a dizer assim:
“Você que passa de carro
Ferre aqui a ver se eu deixo
Venha cá que eu já o agarro
Dou-lhe um pontapé no queixo.”

“Eu durmo sozinha à noite
Vou dormir à beira rio à noite, à noite
Acocorada com o frio à noite, à noite”

Etelvina já cansada de viver sem ninguém
A não ser de vez em quando amores de vai e vem
Pôs um anúncio no jornal a dizer assim:
“Mulher desembaraçada
Quer viver com alma irmã
De quem não seja criada
De quem não seja mamã.”

Etelvina já sabia que não ia encontrar
Nem um príncipe encantado nem um lobo do mar
Só alguém com quem pudesse dizer assim:
“O amor já não é cego
Abre os olhinhos à gente
Faz lutar com mais apego
A quem quer vida diferente.”

O seu homem encontrou-o à noite
A dormir à beira rio, à noite, à noite
Acocorado com frio à noite, à noite.

*Sérgio Godinho no álbum *À queima-roupa* (1974)*