INFORME FINAL DEL PROYECTO

Fundación CNSE
Para la supresión de las barreras de comunicación

Grupo de Tecnología del Habla (GTH). Departamento de Ingeniería Electrónica. ETSI Telecomunicación. Universidad Politécnica de Madrid.

INFORME 2009

Versión: V1.1

Fecha: 5 de marzo de 2009.

Índice

1. Introducción

Este documento es el informe final del proyecto que tiene como título: DESARROLLO DE UN SISTEMA DE TRADUCCIÓN DE VOZ A LENGUA DE SIGNOS ESPAÑOLA PARA UN SERVICIO PÚBLICO DE ATENCIÓN PERSONAL Exp Nº: PAV-070000-2007-567. El objetivo principal del proyecto ha sido el diseño, desarrollo y evaluación de una arquitectura software que permita la traducción de voz a Lengua de Signos Española (LSE). En esta arquitectura, las traducciones a LSE se representan mediante un agente animado en 3D. Con el fin de evaluar y probar la arquitectura planteada se han desarrollado dos demostradores aplicados a la traducción de las frases que un funcionario público pronuncia cuando atiende personalmente a una persona sorda. Estos demostradores están orientados a dos servicios públicos de atención concretos: el servicio de información y gestión para la renovación del Documento Nacional de Identidad (DNI) y el servicio de renovación del carné de conducir en la Dirección General de Tráfico (DGT).

Los principales resultados del proyecto han sido:

- Arquitectura software de traducción de voz a LSE.
- Dos demostradores que permiten la traducción de voz a LSE aplicados a traducir las frases de un funcionario en servicios públicos de atención personal.
- Un entorno de edición para la generación de signos con un agente animado. Dicho entorno permite la traducción de un signo (representado en alguno de los estándares de signo-escritura considerados) a los movimientos del agente animado. Este entorno incorpora un sistema de conversión entre los dos estándares de signoescritura: SEA (Sistema de Escritura Alfabética) y HamNoSys (estándar propuesto desde la Universidad de Hamburgo)

Los principales contenidos multimedia generados:

- Un corpus paralelo con las frases en castellano y su traducción en LSE. Además se incluyen vídeos grabados para una de las frases con el fin de especificar los signos concretos en LSE.
- Generación de una base de datos de contenidos multimedia con las descripciones, características, significado y animación de los signos correspondientes a los servicios públicos de atención abordados en este proyecto: renovación del DNI y del carné de conducir.

Los documentos generados:

- Actual informe donde se recogen todos los detalles del proyecto, de las aplicaciones desarrolladas y los contenidos digitales generados.
- Manuales de usuario de todos los programas generados.
- Documento con las normas de etiquetado en glosas de los corpora.

- Documentación para la impartición de cursos del estándar de signo-escritura HamNoSys.
- Descripción de los contenidos digitales generados.

2. Estado del Arte

2.1. Estudios en Lengua de Signos

La Lengua de Signos es una lengua abierta y flexible, que presenta una gran variabilidad dependiendo de cada país, e incluso, de cada ciudad o región. Es por ello que en los últimos años la Lengua de Signos ha sido objeto de múltiples estudios lingüísticos, que tratan de comprender su estructura, gramática y representación.

En 1960, William Stokoe, profesor de literatura y miembro del Laboratorio de Investigación Lingüística de Washington, publicaba el estudio "Sign Language Structure: An outline of the visual communication system of the American Deaf" [1]. Éste es el primer estudio realizado en Lengua de Signos, concretamente, en Lengua de Signos Norteamericana (ASL – "American Sign Language"). A partir de este momento comienza una trayectoria de estudios de diversos aspectos de la ASL: gramática y estructura, proceso de aprendizaje, representación de los signos, aspectos sociolingüísticos, etc. ([2], [3], [4], [5], [6], [7], [8]). Estos estudios estimularían el desarrollo de trabajos similares sobre las Lenguas de Signos utilizadas en países europeos, como Gran Bretaña ([10], [11], [12]), Noruega [9], Bélgica [13], Dinamarca ([14], [15]), o Francia [16], asiáticos, como Arabia [17], Israel [18] o Japón ([19], [20]), y africanos, como Sudáfrica [22] o Ghana [21]. Así se muestra que la perspectiva de estudio de una Lengua de Signos no debe limitarse al sistema de signos en sí, sino que ha de tener en cuenta contextos más amplios que la expresión lingüística, como la cultura y costumbres de las diferentes comunidades sociales que forman las personas sordas.

En Norteamérica, esta curiosidad científica ha ido acompañada de un reconocimiento, aunque más lento y de manera desigual, del ASL como la lengua materna de la comunidad sorda.

Sin embargo, en el caso de España, la educación del niño sordo ha seguido la tradición del oralismo, entendido como enseñanza del lenguaje verbal exclusivamente, de forma que la Lengua de Signos no ha sido objeto de muchos estudios. En 1991, M.A. Rodríguez [23] realizó un análisis detallado de la Lengua de Signos Española, mostrando sus características principales. Este trabajo ha sido ampliado en otros estudios posteriores ([24], [25], [26]).

2.2. Sistemas de Traducción Automática

Los sistemas de traducción automática pueden clasificarse en dos grandes grupos: sistemas de traducción basada en reglas (directos, por transferencia o por interlingua), desarrollados principalmente a partir de 1970, y sistemas de traducción estadística, que surgen en la década de 1990. Estas distintas aproximaciones de diseño para los sistemas de traducción automática serán descritas más adelante, mientras que en este apartado se realiza un breve estudio de la situación de los sistemas de traducción más importantes que existen actualmente.

Sin duda, el más relevante de los sistemas de traducción automática nunca diseñados ha sido SYSTRAN, sistema basado en reglas desarrollado por Peter Toma en la Universidad de Georgetown a finales de la década de los cincuenta. En 1970 la Comisión Europea adquirió la licencia de SYSTRAN y comenzó a desarrollar traducciones en varios pares de lenguas: francés-inglés, inglés-italiano, e inglés-alemán, entre otros. Se decidió llevar a cabo un ambicioso proyecto de investigación para desarrollar un sistema multilingüe para todas las lenguas de la Comunidad Europea, implicando a todos los grupos de investigación de los estados miembros, dando así lugar al proyecto Eurotra, que utiliza SYSTRAN además de otros sistemas de traducción [27]. En 1997 el portal de Internet Altavista llegó a un acuerdo con SYSTRAN para ofrecer un servicio de traducción gratuito por web, BABELFISH. En este momento, SYSTRAN es el sistema de traducción más desarrollado y utilizado, con 35 pares de lenguas disponibles [28].

En 1967, la Universidad de Saarbrücken comenzó un proyecto de investigación para explorar la posibilidad de utilizar SYSTRAN para la traducción del ruso al alemán. El fracaso del proyecto hizo que empezaran a trabajar en un nuevo prototipo de sistema de traducción, dando lugar finalmente, en 1972, al sistema multilingüe basado en transferencia conocido como SUSY. Las lenguas implicadas en este sistema son principalmente alemán, ruso, inglés y francés. En los años siguientes, se realizaron múltiples versiones de SUSY, cada vez más influenciadas por el proyecto Eurotra hasta que en 1986, ambas investigaciones se fusionaron.

Tras SYSTRAN, el segundo sistema en veteranía e importancia es sin duda el sistema de traducción por transferencia METAL. Fue diseñado en 1961 en el LRC (“Linguistic Research Center” – Centro de Investigación Lingüística) de la Universidad de Texas, bajo la dirección de Winfred Lehmann, para el par de lenguas inglés-alemán. Al par inicial se añadieron pronto otros siete, con las principales lenguas europeas: francés, holandés, danés y español. Tras pasar por empresas como SIEMENS o el grupo belga Lernout & Hauspie, actualmente los distintos desarrollos de METAL dependen de la empresa Sail Labs.

Siguiendo la línea de desarrollo de sistemas de traducción por transferencia, en 1976 surge Météo, sistema desarrollado por el grupo TAUM (“Traduction Automatique de l´Univesité de Montréal” – Traducción Automática de la Universidad de Montreal) para traducir boletines del tiempo atmosférico del inglés al francés. El sistema continúa actualmente en operación, y su éxito se basa fundamentalmente en la precisión que se obtiene en la traducción al restringir el dominio de aplicación al lenguaje propio de las previsiones meteorológicas.

Durante la década de los ochenta, al diseño de sistemas de traducción por transferencia se añaden nuevos elementos basándose en la idea de sistemas interlingua. Surgen así los sistemas DLT (“Distributed Language Translation” – Traducción de Lenguaje Distribuido), en Utrecht, basado en una modificación del Esperanto, y Rosetta, diseñado por Phillips.

Por otro lado, desde mediados de los ochenta, IBM ha centrado sus esfuerzos de traducción automática en el proyecto LTM, dirigido por Michael Cord y desarrollado simultáneamente en los laboratorios de EEUU, España, Israel y Alemania. Se han desarrollado traducciones de doce pares de lenguas y en la actualidad se encuentra disponible con el nombre WebSphere.

Desde comienzos de la década de los noventa, la mayoría de los programas de traducción se han adaptado al ordenador personal. Dos de los primeros en hacerlo fueron PC-Translator (de Linguistic Products) y Power Translator (de Globalink). Cabe destacar también los sistemas TransLinGo, de Fujitsu, LogoVista, Tsunami y Typhoon, para traducción de japonés a otros idiomas, y ProMT y PARS, para la traducción del ruso a otras lenguas europeas. Además de estos sistemas comerciales, también se han desarrollado otros sistemas de uso interno, para determinadas organizaciones, entre los que cabe destacar los sistemas SPANAM (español-inglés) y ENGSPAN (inglés-español), desarrollados por la Organización de la Salud Panamericana.

La generalización de Internet a partir de 1995 ha permitido poner en marcha servicios de traducción automática a través de la web. Este hecho permite a los fabricantes ofrecer a los clientes potenciales el acceso a versiones de demostración temporales o parciales, fáciles de conseguir e instalar. Por otro lado, hace posible que cualquier usuario con acceso a Internet pueda probar las posibilidades de traducción automática en alguno de los numerosos sitios que ofrecen servicios abiertos y gratuitos, algunos de los cuales se indican a continuación. Estos sistemas de traducción suelen ser basados en reglas, con una cobertura amplia, a costa de una calidad generalmente muy baja.

- BABELFISH: http://babelfish.altavista.com (licencia de SYSTRAN).
- SYSTRAN: http://www.systransoft.com
- GOOGLE: http://translate.google.com
- ProMT: http://www.translate.ru/
- FreeTRANSLATION: http://ets.freetranslation.com:5081/
- VOILA: http://www.voila.fr/ (licencia de Softissimo).
- EL Mundo: http://www.el-mundo.es/traductor/ (licencia de Softissimo).
- NeuroTrans: http://www.tranexp.hr
- InterTran (tm): http://www.tranexp.com
- World Language: http://www.worldlanguage.com/Translation.htm (de InterTran).
- Sail Labs: http://t1.sail-labs.com/t1probe.html (anteriormente METAL).

- AutomaticTrans: http://www.automatictrans.es (español, catalán, portugués).
- LogoVista: http://www.logovista.jp (traductor del inglés al japonés).
- TARJIM: http://www.tarjim.com (traductor del inglés al árabe).

Todos los sistemas comentados anteriormente son sistemas de traducción basados en reglas. En los años noventa empezaron a desarrollarse sistemas de traducción automática estadística, entre los que cabe destacar el sistema CANDIDE, desarrollado por el grupo del Thomas J. Watson Center de IBM en Nueva York. Este sistema no llegó a comercializarse, pero los buenos resultados obtenidos supusieron un hito histórico en la reorientación de las investigaciones.

Entre las aportaciones más recientes a dicho sistema, cabe destacar la del grupo ISI ("Information Sciences Institute" – Instituto de las Ciencias de la Información) de la Universidad Southern California, que desarrolla EGYPT, un paquete de software para desarrollar sistemas basados en la estadística a partir de corpora bilingüe. EGYPT a su vez tiene un kit de herramientas que incorpora el sistema GIZA, que consiste en un programa de entrenamiento que aprende modelos de traducción estadística a partir de corpora bilingües.

También colaboración del grupo ISI es PHARAO, desarrollado por Philipp Koehn. Este sistema necesita un modelo de traducción y modelo de lenguaje para realizar la traducción de una lengua a otra.

Por último, es importante destacar que en los últimos años han surgido otras alternativas de traducción. Los avances generados en el reconocimiento y síntesis del habla han permitido el desarrollo de proyectos de traducción automática del habla, principalmente en Gran Bretaña y en Japón. La traducción del habla natural es una de las principales áreas de investigación en proyectos como C-Star, ATR ("Advanced Telecommunications Research" – Investigación de Telecomunicaciones Avanzadas), Vermobil, Eutrans, LC-Star, PF-Star, y TC-Star. Excepto este último, todos son proyectos enfocados a la traducción automática del habla en dominios restringidos (por ejemplo, una agencia de viajes), con vocabularios de tamaño medio. Los mejores sistemas de traducción desarrollados están basados en soluciones estadísticas [29], incluyendo técnicas basadas en ejemplos [30], traductores de estados finitos [31] y otras soluciones basadas en datos. Los importantes progresos conseguidos en la traducción del habla se deben principalmente a factores como la aparición de medidas de error [32], la mejora de la eficiencia de los algoritmos de entrenamiento [33], el desarrollo de modelos dependientes de contexto [30], y el desarrollo de algoritmos de generación eficientes [34].

2.3. Agentes Animados Virtuales

En los últimos años, el campo de la animación ha experimentado un gran desarrollo, que se debe en gran parte al interés suscitado por la industria de los videojuegos. Es por esta razón que una parte importante de la comunidad científica dedicada al estudio de las tecnologías del habla está tratando de

desarrollar y evaluar agentes virtuales embebidos en sistemas de lenguaje hablado.

Estos sistemas proporcionan una gran variedad de servicios en distintos ámbitos. En el MIT (Instituto Tecnológico de Massachussets) se ha desarrollado un agente animado embebido en una aplicación, MACK ("Media Lab Autonomous Conversational Kiosk"), que puede responder preguntas acerca de los distintos grupos de investigación de Media Lab, así como de sus proyectos e investigadores [35].

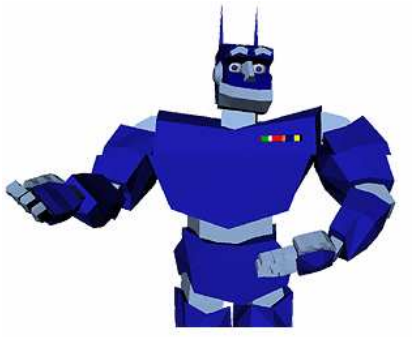


Figura 1:Agente animado MACK

En KTH (Universidad Tecnológica Real) de Estocolmo se han desarrollado varios sistemas de diálogo multimodo, que ofrecen información tanto escrita como hablada, donde se incorporan agentes animados para mejorar el interfaz ([36], [37]). Entre estos sistemas se encuentran Waxholm [38], un sistema de planificación de viajes para barcos en el archipiélago de Estocolmo, August [39], un sistema de información del Centro Cultural de Estocolmo (0), y AdApt [40], sistema en que el usuario interactúa con un agente virtual animado para encontrar apartamentos en Estocolmo.

Figura 2: Sistema August

Otra aplicación que combina el lenguaje con las tecnologías de animación ha sido el desarrollo de libros interactivos para el aprendizaje. En el CSLU ("Center for Spoken Language Understanding" – Centro para la Comprensión del Lenguaje Hablado) del Instituto de Postgrado de Oregón ("Oregon Graduate Institute", OGI) se ha desarrollado una herramienta ([41], [42]) que integra un agente virtual animado llamado Baldi, y permite desarrollar libros interactivos rápidamente, con recursos multimedia e interacción natural. Actualmente se está ampliando esta herramienta en el CSLR ("Center for Spoken Language

Research" – Centro para la Investigación del Lenguaje Hablado) [43] de la Universidad de Colorado.

Figura 3: Agente Baldi

En la actualidad son muchas las herramientas disponibles para el diseño de objetos en 3D. A partir de ellas es posible diseñar agentes virtuales animados, empleando una u otra herramienta en función de la aplicación y del grado de detalle y realismo necesarios. Algunas de estas herramientas son 3D Studio Max, desarrollado por Autodesk Media & Entertainment, y que permite crear tanto modelados como animaciones en 3D a partir de una serie de vistas de plantas y alzados, Lightwave 3D, desarrollado por NewTek, Cool 3D, que ofrece la posibilidad de crear objetos tridimensionales a partir de formas predefinidas, Toon 3D, desarrollado por Nik Lever, y diseñado como una herramienta de desarrollo interactiva, que permite la creación de juegos, demos y animaciones para su distribución en la web, y por último, It´s Me (actualmente, iClone), diseñado para aplicaciones de baja complejidad, que permite importar imágenes del usuario y adaptarlas a caracteres predefinidos para la creación de escenas y movimientos [44].

2.4. Sistemas de Traducción de Voz a Lengua de Signos

En los últimos años se han desarrollado sistemas de traducción de voz o texto a la Lengua de Signos, con el fin de eliminar las barreras de comunicación que afectan a las personas sordas. En este apartado se repasa el estado del arte de este tipo de sistemas, la mayoría de los cuales se dedican a la Lengua de Signos Americana (ASL – "American Sign Language").

En primer lugar, existen paquetes ya comercializados de sistemas de traducción del inglés al ASL. El sistema de Aramedia incorpora un traductor unidireccional de inglés a Lengua de Signos, que además muestra la representación de cada uno de los signos que surgen de la traducción. Contiene más de 2500 videos de signos, para 4500 palabras inglesas [45].

Figura 4: Sistema Aramedia de traducción de inglés a ASL

Por otro lado, en 1999, en Connecticut, surge la empresa SignTel, formada por personas oyentes y sordas, que ha desarrollado la aplicación SignTel Interpreter [49]. Esta aplicación permite traducir el inglés, ya sea hablado o escrito, a la ASL. Dispone de más de 30.000 palabras y frases diferentes, y sus correspondientes representaciones en la Lengua de Signos, capturadas en video y digitalizadas. El interfaz de la aplicación se muestra en la siguiente figura.

Figura 5: Interfaz de la aplicación SignTel

Otro sistema destacable es la herramienta SignSmith Studio [50], de la empresa VCom3D, que permite traducir textos del inglés a la ASL, a través de agentes animados virtuales generados con el software SigningAvatar. Este software permite crear agentes animados en 3D para la representación de la parte manual de los signos y de sus expresiones faciales. El sistema tiene una base de datos de más de 10.000 palabras de la lengua inglesa y está diseñado principalmente para la traducción de contenido web.

Fuera ya del ámbito de empresa, el proyecto europeo eSIGN (“Essential Sign Language Information on Government Networks” - Información Esencial en Lengua de Signos en Redes Gubernamentales), desarrollado en el Instituto de Lengua de Signos Alemana de la Universidad de Hamburgo, es actualmente uno de los más importantes en el campo del desarrollo de herramientas para la generación automática de contenido en Lengua de Signos. En este proyecto se ha desarrollado el agente animado Virtual Guido para la representación de los signos. Se dispone además de un entorno que permite crear los signos de una forma sencilla, flexible y rápida. Las herramientas desarrolladas en este proyecto se orientan principalmente a la traducción del contenido de páginas web a la Lengua de Signos. Se está utilizando este proyecto en páginas web de Alemania, Holanda y Reino Unido [46], ofreciendo traducciones a las Lenguas de Signos Alemana, Holandesa e Inglesa, respectivamente.

Figura 6: Agente animado VGuido

Se han desarrollado también otros sistemas de traducción que utilizan agentes animados virtuales para la representación de los signos. En la Escuela de Telecomunicaciones de la Universidad de DePaul, en Chicago, se ha desarrollado un sistema de traducción del inglés hablado a la ASL, para su uso en aeropuertos [47]. El sistema permite a los guardias de seguridad comunicarse con los pasajeros sordos a través del agente animado Paula, de forma que la persona sorda pueda tener acceso a la información acerca de los procesos de seguridad.

Figura 7: Agente animado Paula

En la Universidad de East Anglia se ha desarrollado el proyecto TESSA ("TExt and Sign Support Assistant" – Asistente de Texto y Signado), englobado en el proyecto ViSiCAST ("Virtual Signing: Capture, Animation, Storage and Trasmission" – Signado Virtual: Captura, Animación, Almacenamiento y Transmisión), en colaboración con la Oficina de Correos de Reino Unido, para hacer posible que los trabajadores de estas oficinas puedan comunicarse con clientes sordos [48]. El sistema convierte la voz a la Lengua de Signos Inglesa (BLS – "British Sign Language"), siendo los signos representados por un agente virtual.

Figura 8: Agente animado proyecto TESSA

En la Universidad de Nuevo México, Angus Grieve-Smith está desarrollando el sistema SignSynth. Esta aplicación permite traducir un texto en lengua inglesa a su correspondiente en ASL, representando los signos a través de un agente animado que puede visualizarse en cualquier navegador web [51]. También en esta misma universidad se ha desarrollado una máquina de traducción del inglés al ASL, en el dominio de aplicación de "información del tiempo atmosférico". Es un sistema basado en reglas por transferencia, y se compone de cuatro bloques: analizador léxico, programa de análisis, módulo de transferencia y módulo de generación [52].

Cabe destacar también el proyecto TEAM ("Translation from English to ASL by Machine"), de la Universidad de Pennsylvania (Philadelphia), en el que se ha desarrollado un sistema de traducción del inglés a la ASL [53]. Éste es un sistema de traducción basado en reglas, que emplea el método de transferencia, y permite representar los signos a través de un agente animado virtual.

Figura 9: Agente animado del sistema TEAM

IBM está desarrollando, en colaboración con la Universidad de East Anglia (UEA) y el RNID ("Royal Nacional Institute for Deaf People" – Instituto Nacional Real para Personas Sordas), el sistema SiSi ("Say It, Sign It"), para la traducción del inglés a la Lengua de Signos Inglesa (BSL). El sistema dispone de un reconocedor del habla, que permite traducir la voz a texto, y un módulo que permite traducir el texto a la Lengua de Signos. Los signos son representados mediante un agente animado, generado con la tecnología de

animación de la UEA. La base de datos de signos ha sido desarrollada por el RNID [54].

En el Centro Nacional para la Tecnología del Lenguaje de la Universidad de Dublín, se está desarrollando un sistema de traducción del inglés a la Lengua de Signos Irlandesa (ISL), para mejorar la accesibilidad de las personas sordas a los avisos e información en aeropuertos (cambios de puerta de embarque, hora de salida, etc.) [55]. El sistema se centra en el método estadístico de traducción, basado en ejemplos, y emplea un agente animado, obtenido del software Poser 4 ProPack, para representar los signos.

Además de estos sistemas, en varios grupos y universidades se han desarrollado o se están desarrollando sistemas de traducción a otras Lenguas de Signos. Así, se trabaja en sistemas de traducción del inglés a la Lengua de Signos Sudafricana ([56], [57]), del tailandés a la Lengua de Signos Tailandesa [58], del checo a la Lengua de Signos Checa [59], o del polaco a la Lengua de Signos Polaca ([60], [61]).

En cuanto a la Lengua de Signos Española (LSE), parece ser que ya se han desarrollado sistemas de traducción automática del LSE al español oral [62]. En la universidad de Zaragoza se está desarrollando un traductor del español al LSE basado en reglas gramaticales y morfológicas [63].

2.5. Referencias

[1] W.C. Stokoe, “Sign Language Structure: An Outline of the Visual Communication Systems of the American Deaf”, Journal of Deaf Studies and Deaf Education, Vol.10, Nº 1, 2005.

[2] U. Bellugi, E. Klima, *“The roots of language in the sign talk of the deaf”*, Psychology today, Vol.6: 661-676, 1972.

[3] R. Hoffmeister, “The acquisition of ASL by deaf children of deaf parents”, Universidad de Minessota, 1977.

[4] R. Wilbur, *“American Sign Language and Sign Systems”*, Baltimore, University Press, 1979.

[5] J.E. Pyers, “Indicating the body: Expression of body part terminology in American Sign Language”, Language Sciences, Junio 2006.

[6] C. Christopoulou, J.D. Bonvillian, *“Sign language, pantomime, and gestural processing in aphasic persons: A review”*, Journal of Communication Disorders, Vol.18, 1985.

[7] S.D. Fischer, P. Siple, *“Theoretical Issues in Sign Language Research”*, University of Chicago Press, 1990.

[8] J. Woodward, “Implications for sociolinguist research among the deaf”, Sign Language Studies, Nº 1, 1972.

[9] M. Vogt-Svedensen, “Mouth position and mouth movement in Norwegian Sign Languages”, Sign Language Studies, Nº 33, 1981.

[10] G.L. Zaitseva, “The Sign Language of the Deaf as a Colloquial System”, Language in Sign, Londres, 1983.

[11] M. Atherton, *“Welsh today BSL tomorrow”*, DeafWorlds, Vol.15, Nº 1, pp.11-15, 1999.

[12] J. Kyle, *"British Sign Language"*, Special Education, Nº 8, pp: 19-23, 1981.

[13] L. Meurant, *"Anaphora, role shift and polyphony in Belgian sign language"*, International Conference on Theoretical Issues in Sign Language Research, Barcelona, Septiembre 2004.

[14] E. Engberg-Pedersen, "From pointing to reference and predication: pointing signs, eyegaze, and head and body orientation in Danish Sign Language", Pointing: Where language, culture and cognition meet. Erlbaum, 2003.

[15] B. Hansen, "Varieties in Danish Sign Language and grammatical features of the original Sign Language", Sign Languages Studies, Vol.8, pp: 249-256, 1975.

[16] P. Olerón, "Le Langage Gestuel des Sourds: Syntaxe et Communication", CNRS, 1978.

[17] M.A. Abdel-Fattah, *"Arabic Sign Language: A perspective"*, Journal of Deaf Studies and Deaf Education, Vol.10, Nº 2, pp: 212-221, 2005.

[18] I.M. Schlesinger, "The Grammar of Sign Language and the Problem of Language Universals", Massachussets, Logos Press, Morton, 1971.

[19] N. Masataka, *"Neural correlates for numerical processing in the manual mode"*, Journal of Deaf Studies and Deaf Education, Vol.11, Nº 2, pp: 144-152, 2006.

[20] M. Notoya, S. Suzuki, M. Furukawa, R. Umeda, *"Method and Acquisition of Sign Language in Profoundly Deaf Infants"*, Japan Journal of Logopedics and Phoniatrics, Nº 27, pp: 235-243, 1986.

[21] V. Nyst, *"Verbs of motion in Adamorobe Sign Language"*, International Conference on Theoretical Issues in Sign Language Research, Barcelona, Septiembre 2004.

[22] C. Penn, R. Lewis, A. Greenstein, *"Sign Language in South Africa: some research and clinical issues"*, South African Disorder of Communication, Vol.31, pp: 6-11, 1984.

[23] M.A. Rodríguez González, *"Lengua de Signos"*, Tesis Doctoral 1996. CNSE y Fundación ONCE. Biblioteca Virtual Miguel de Cervantes.

[24] B. Gallardo, "Estudios Lingüísticos sobre la lengua de signos española", Universidad de Valencia, Ed. Agapea, 2002.

[25] A. Herrero Blanco, V. Salazar García, *"Non-verbal predicability and copula support rule in Spanish Sign Language"*, Morphosyntactic expression in functional grammar (Functional Grammar Series), Vol.27, pp: 281-315, 2005.

[26] I. Reyes, "Comunicar a través del silencio: las posibilidades de la lengua de signos española", Universidad de Sevilla, 2005.

[27] W.J. Hutchins, H.L. Somers, *"An Introduction to Machine Translation"*, Academic Press, 1992.
http://www.hutchinsweb.me.uk/

[28] J. Abaitua, *"Introducción a la traducción automática"*, Grupo DELi, Universidad de Deusto, 2002.

[29] J. Och, H. Ney, *"Discriminative Training and Maximum Entropy Models for Statistical Machine Translation"*, Anual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, pp: 295-302, 2002.

[30] E. Sumita, Y. Akiba, T. Doi et al, *"A Corpus-Centered Approach to Spoken Language Translation"*, Conference of the Association for Computational Linguistics (ACL), Hungría, pp: 171-174, 2003.

[31] F. Casacuberta, E. Vidal, *"Machine Translation with Inferred Stochastic Finite-State Transducers"*, Computacional Linguistics, Vol. 30, Nº 2, pp: 205-225, 2004.

[32] K. Papineni, S. Roukos, T. Ward, W.J. Zhu, *"BLEU: a method for automatic evaluation of machine translation"*, 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, pp: 311-318, 2002.

[33] J. Och, H. Ney, *"A Systematic Comparison of Various Statistical Alignment Models"*, Computational Linguistics, Vol. 29, Nº 1, pp: 19-51, 2003.

[34] P. Koehn, F.J. Och, D. Marcu, *"Statistical Phrase-based translation"*, Human Language Technology Conference, Edmonton, Canadá, pp: 127-133, Mayo 2003.

[35] J. Cassell, T. Stocky, T. Bickmore, Y. Gao, Y. Nakano, K. Ryokai, D. Tversky, C. Vaucelle, Vilhjálmsson, *"MACK: Media lab Autonomous Conversational Kiosk"*, Proceedings of Imagina: Intelligent Autonomous Agents, Monte Carlo, Mónaco, 2002.

[36] J. Gustafson, *"Developing multimodal spoken dialogue systems- Empirical studies of spoken human-computer interactions"*, Department of Speech, Music and Hearing, Royal Institute of Technology, Estocolmo, Suecia, 2002.

[37] B. Granström, D. House, J. Beskow, *"Speech and Gestures for Talking Faces in Conversational Dialogue Systems"*, Multimodality in Language and Speech Systems, Kluwer Academic Publishers, pp: 209-241, 2002.

[38] J. Bertenstam, *"The Waxholm system – A progress report"*, Proceedings on Spoken Dialogue Systems, Vigso, Dinamarca, 1995.

[39] M. Lundeberg, J. Beskow, *"Developing a 3D-agent for the August dialogue system"*, Proceedings on Audio-Visual Speech Processing, Santa Cruz, 1999.

[40] J. Gustafson, L. Bell, *"Speech technology on trial: Experiences from the August system"*, Journal of Natural Language Engineering: Special Issue on Best Practice in Spoken Dialogue Systems, pp. 273-286, 2003.

[41] S. Sutton, R. Cole, *"Universal speech tools: the CSLU toolkit"*, Proceedings on Spoken Language Processing, Sydney, Australia, pp: 3221-3224, 1998.

[42] R. Cole, "New Tools for interactive speech and language training: Using animated conversational agents in the classrooms of profoundly deaf children", Proceedings ESCA/SOCRATES Workshop on Method and Tool Innovations for Speech Science Education, London, pp: 45-52, 1999.

[43] R. Cole, S. Van Vuuren, B. Pellom, K. Hacioglu, J. Ma, J. Movellan, S. Schwartz, D. Wade-Stein, W. Ward, J. Yan, *"Perceptive Animated Interfaces: First Steps Toward a New Paradigm for Human Computer Interaction"*, IEEE Transactions on Multimedia: Special Issue on Human Computer Interaction, Vol. 91, Nº. 9, pp. 1391-1405, 2003.

[44] A. Huerta, *"Entorno de desarrollo para la creación de animaciones con agentes virtuales 3D"*, Proyecto Fin de Carrera, Grupo de Tecnología del Habla (GTH), Departamento de Ingeniería Electrónica, Universidad Politécnica de Madrid, Diciembre 2006.

[45] Aramedia, Instant Immersion American Sign Language Express v2.0 http://aramedia.com

[46] Proyecto Europeo eSIGN. http://www.sign-lang.uni-hamburg.de/eSIGN/

[47] Proyecto de la Universidad DePaul. http://asl.cti.depaul.edu/

[48] Proyecto TESSA. http://www.visicast.cmp.uea.ac.uk/Tessa.htm

[49] SignTel Inc. http://www.signtelinc.com/

[50] VCom3D Inc. http://www.vcom3d.com/signsmith.php

[51] A.B. Grieve-Smith, "SignSynth: A Sign Language Synthesis Application Using Web3D and Perl", Wachsmuth and T. Sowa, pp: 134-145, 2002.

[52] A.B. Grieve-Smith, *"English to American Sign Language Machine Translation of Weather Reports"*, Proceedings of the Second High Desert Student Conference in Linguistics, Universidad de Nuevo México, 1999.

[53] L. Zhao, K. Kipper, W. Schuler, C. Vogler, N. Badler, M. Palmer, *"A Machine Translation System from English to American Sign Language"*, J.S. White (Ed.), pp: 54-67, 2000.

[54] Artículo de prensa "IBM Research Demonstrates Innovative Speech to Sign Language Translation System", 12 Septiembre 2007. Disponible en web: http://www-03.ibm.com/press/us/en/pressrelease/22316.wss

[55] S. Morrissey, A. Way, *"Joining Hands: Developing a Sign Language Machine Translation System with and for the Deaf Community"*, Conference & Workshop on Assistive Technologies for People with Vision & Hearing Impairments, Assistive Technology for All Ages, 2007.

[56] L. van Zijl, D. Barker, *"South African Sign Language Machine Translation System"*, Association for Computing Machinery, 2003.

[57] L. van Zijl, A. Combrink, "The South African Sign Language Machine Translation Project: Issues on Non-manual Sign Generation", SAICSIT, 2006.

[58] S. Dangsaart, N. Cercone, K. Naruedomkul, B. Sirinaovakul, *"Bridging the gap: Thai – Thai Sign Machine Translation"*, Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pp: 191-199, 2007.

[59] J. Kanis, J. Zahradil, F. Jurcícek, L. Müller, *"Czech-Sign Speech Corpus for Semantic Based Machine Translation"*, P. Sojka, I. Kopecek, K. Pala (Eds.), pp: 613-620, 2006.

[60] N. Suszczanska, P. Szmal, J. Francik, *"Translating Polish Texts into Sign Language in the TGT System"*, Applied Informatics, 2002.

[61] J. Francik, P. Fabian, *"Animating Sign Language in the Real Time"*, 20th IASTED International Multi-Conference, Applied Informatics, pp: 276-281, 2002.

[62] J. Moreno del Pozo, J.M. León, A.M. Silva, *"Sistemas de Traducción Automática del Lenguaje de Signos Español al Español Oral"*, Lengua y Tecnologías de la Información, 1998.

[63] S. Baldassarri, F.J. Royo, *"Traductor de Español a LSE basado en reglas gramaticales y morfológicas"*, VIII Congreso de Interacción Persona-Ordenador, 2007.

3. Estudio lingüístico

En esta tarea se ha realizado el estudio lingüístico de las frases pronunciadas por el funcionario público en los dos servicios propuestos (o dominios de aplicación). Este estudio lingüístico ha supuesto principalmente la recopilación de las frases que con mayor frecuencia pronuncian los funcionarios en los dos servicios de atención personal y su traducción posterior a LSE.

3.1. Base de datos para la renovación del DNI.

La base de datos se compone de dos partes, una corresponde a las oraciones en lengua castellana, y la otra corresponde a las oraciones en lengua de signos, equivalentes una a otra en cuanto a significado. Para generar la base de datos se han realizado tres pasos: el primero corresponde a la selección del conjunto de frases en castellano pertenecientes al ámbito de trabajo; el segundo corresponde a la traducción de estas frases a LSE (representando las frases en glosas) por expertos; por último, se han realizado grabaciones de todas las representaciones en LSE de las frases, también por expertos.

- **Selección de las oraciones en castellano:** las frases seleccionadas han sido generadas por varias personas. Algunas de las frases de la base de datos ya se utilizaron en bases de datos de anteriores proyectos. Posteriormente se fue ampliando esta selección por miembros del grupo de investigación, basándose en la propia experiencia en los aspectos burocráticos del DNI. En cuanto a las frases seleccionadas de las bases de datos anteriores, cierto número de ellas fueron obtenidas mediante entrevistas con funcionarios de la Administración, añadiendo así frases típicas del día a día en este tipo de procesos burocráticos.

- **Traducción de las oraciones a LSE:** todas las frases en castellano seleccionadas fueron traducidas por personas sordas conocedoras de la lengua castellana, intentando mantener lo más fiel posible el significado contenido en las oraciones originales.

- **Grabación de los videos:** todas las frases han sido representadas por expertos en la LSE, para poder comprobar en todo momento qué signo es el etiquetado por cada una de las glosas que están representadas en las traducciones de las frases.

Todas las traducciones se realizaron de forma escrita, forma en la que también se guardan en la base de datos, representando cada uno de los signos mediante glosas. La mayoría de las personas sordas tienen grandes dificultades para comprender la lengua castellana escrita, siendo muy difícil que la comprensión llegue a ser absoluta, principalmente en personas sordas prelocutivas. Además, hay que tener en cuenta que la representación de los signos en glosas (mediante reglas de etiquetado) se encuentra en proceso de normalización, por lo que no hay unas reglas fijas de etiquetado todavía. Por

ello, en las traducciones obtenidas por personas sordas, a pesar de su alto conocimiento de la lengua castellana escrita, tuvieron ciertos problemas en la representación de las glosas, habiendo algunos errores ortográficos. Además de obtener la traducción en forma de glosas de las oraciones en castellano, hubo que revisar ortográficamente las glosas de cada traducción, ya que cualquier error podía inducir uno mayor en la traducción final, ya que debe existir unicidad con la base de datos de signos (en la que también se utiliza la representación por glosas, por lo que es de vital importancia que no existan diferencias entre unas y otras).

En algunos casos también se encontraron problemas en la unicidad de bases de datos debido a la poca estandarización existente en cuanto a la escritura de signos. Esto hace que no sólo no haya un estándar único de escritura, sino que además, muchas de las representaciones del mismo signo con el mismo tipo de escritura (nos referimos principalmente a las glosas) son distintas, dependiendo del ámbito del que se obtenga. De esta manera ocurrió que encontramos diferentes glosas para representar el mismo signo (como por ejemplo en el caso de FOTO y FOTOGRAFÍA, cuya representación es la misma), ya que en ciertas ocasiones éstas eran representadas de una manera o de otra. Para conseguir la unicidad entre bases de datos hubo que revisar estas diferencias, para conseguir en toda la aplicación representar cada signo de una única manera.

También se han encontrado problemas por el etiquetado que realizan las glosas. Hay casos en los que existen diferentes signos (representaciones naturales) que se etiquetan mediante la misma glosa (equivalencia en el castellano a una palabra con varios significados). Esto genera un problema en la elección de signos, principalmente en la búsqueda por glosas, ya que se necesita unicidad en el etiquetado. Esto ocurre, por ejemplo, con la glosa HABER. Existen dos signos que se etiquetan con esta glosa, uno indicando obligación (“haber que”) y el otro existencia (“hay una mesa”). Por unicidad y para no generar posibles equívocos, decidimos cambiar el signo de obligación, sustituyéndolo por DEBER, dejando así un único signo con la etiqueta HABER.

Dentro de los plurales encontramos varios casos especiales. Algunos de los plurales (en este caso nos referimos a plural entendido en castellano, cuya representación se plasma en la glosa correspondiente, sin embargo el tratamiento del signo es distinto) se representan con el mismo signo que representa el singular (en nuestro contexto de aplicación). Tenemos el caso de las siguientes glosas:

- **APELLIDO Vs APELLIDOS:**

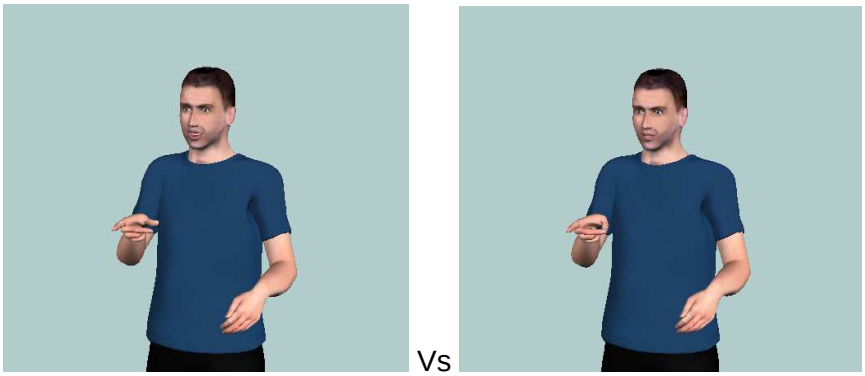


Figura 10: Representación del signo *APELLIDO* Vs *APELLIDOS*

- **AÑO:**

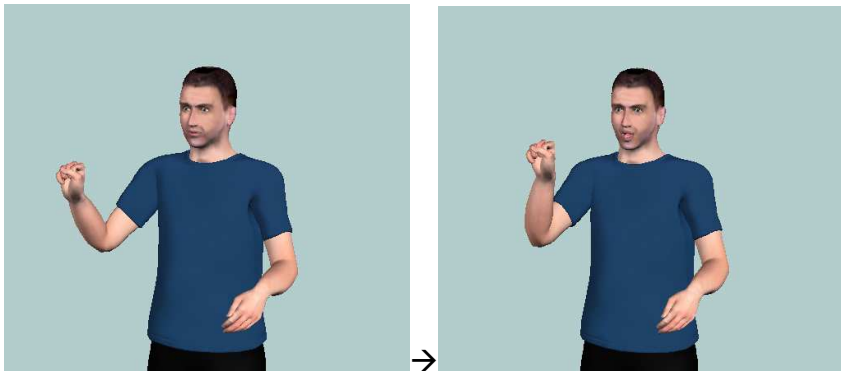


Figura 11: Representación del signo *AÑO*

- **AÑOS:**

Figura 12: Representación del signo *AÑOS*

Podemos comprobar que en el caso de APELLIDO/APELLIDOS, el signo que representa las dos glosas es el mismo, mientras que para AÑO y para AÑOS, cada glosa requiere de un signo diferente (tienen distintos significados). En los casos en que los plurales se representan con el mismo signo que el singular, tendría sentido en la base de datos escoger siempre una de las glosas para simplificar y dar más sentido de unicidad. Sin embargo, en estos casos, y teniendo en cuenta que esta base de datos únicamente se utiliza para el

método de traducción basado en ejemplos, la diferencia de glosas puede ayudar en la propia traducción.

Este tipo de casos ayudaría siempre que existan dos ejemplos (uno utilizando el plural y otro el singular), bien diferenciados en su estructura sintáctica y significado (se refiere al de toda la oración), con lo que la diferenciación de glosas ayudaría en el proceso de traducción (primer paso de la traducción, método basado en ejemplos), guiando hacia una o hacia otra. Sin embargo, en casos en los que la estructura sintáctica y el significado no difieren demasiado, sería preciso unificar, ya sea dejando una única glosa en las bases de datos, o categorizándolas (este tema se discutirá en el capítulo dedicado a la traducción, donde también se tratará más en profundidad el tema de los plurales).

También existen casos en los que se han obtenido muchas glosas representadas por el mismo signo (se han obtenido al realizar las traducciones). Esto también es resultado de la falta de estandarización y de convenios, produciendo distintas glosas (diferencias en castellano, aunque el significado en el ámbito de trabajo sea el mismo, lo que también se refleja en que se representen con el mismo signo), como en el caso siguiente:

- **DOCUMENTO/PAPEL/DOCUMENTACIÓN:**

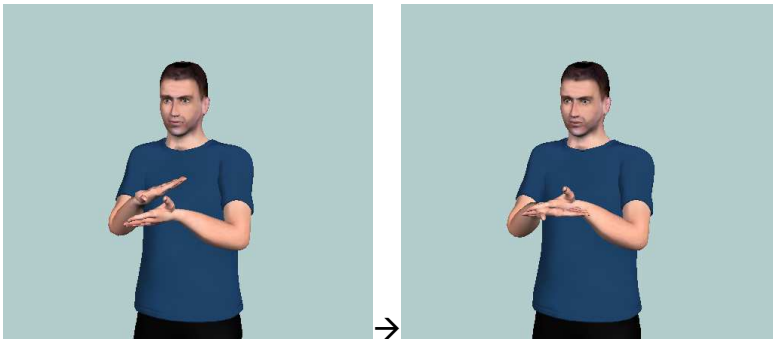


Figura 13: Representación del signo *DOCUMENTO-PAPEL-DOCUMENTACIÓN*

El signo que representa todas estas glosas en el ámbito de trabajo de la aplicación es el mismo, lo que también genera un problema de unicidad en las bases de datos. Por ellos, se encuentra la misma discusión de unicidad y posible categorización de todas estas glosas. Equivalente al caso anterior, tendrá sentido si las estructuras sintácticas y el significado difieren, guiando el proceso de traducción a una u otra oración. También tiene sentido diferenciar las glosas en el caso de que se realice una búsqueda de signos por glosas, pudiendo encontrar el signo deseado de más de una manera (la búsqueda de signos por glosas se especifica en el capítulo cuarto).

Como hemos comentado, una de las principales causas de los problemas descritos anteriormente es la poca normalización del etiquetado de los signos (representación en glosas). Este etiquetado todavía se encuentra en proceso abierto de estandarización, por lo que muchos de los problemas existentes provienen de este hecho. En el tercer apartado (3.3) se encuentran descritas

las reglas de generación utilizadas durante el proyecto para etiquetar los signos.

Una vez comentados los problemas encontrados en la generación de la base de datos de ejemplos, pasaremos a detallar su contenido. Ésta se encuentra en un fichero de tipo Excel, aunque su uso en la aplicación no lo utilice como tal, sino que debe ser tratado para poder ser utilizado por ésta (el tratamiento exacto de la base de datos por la aplicación se describe en el capítulo dedicado a la traducción, correspondiente al quinto capítulo). El fichero en el que se guardan todas las frases es: frases_DNI.xls. Dentro de este fichero de tipo Excel encontramos dos hojas, diferenciando el conjunto de frases original (de la anterior base de datos), de las frases seleccionadas por los miembros del grupo de investigación (Ampliación por el GTH). En cada una de estas hojas la división y estructura de las frases es la misma.

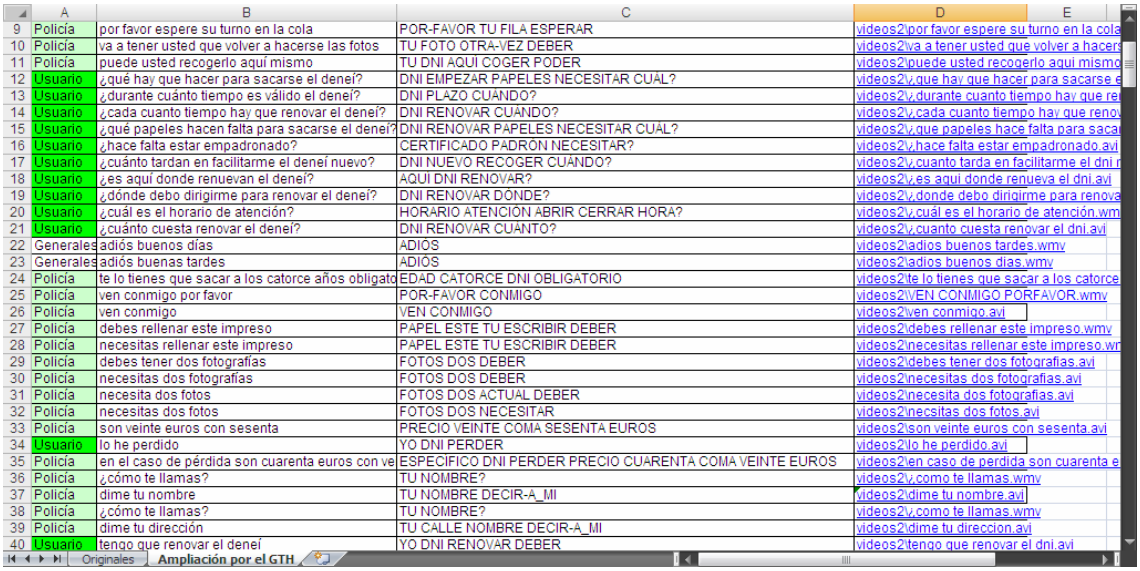
En cada tabla (hoja) encontramos cuatro campos, cuya descripción es la siguiente:

- **TIPO:** en este campo se indica la procedencia de las frases, pudiendo diferenciar tres tipos: *Generales* (son el tipo de frases utilizadas tanto por el usuario como por el funcionario), *Policía* (frases utilizadas por el funcionario de la Administración) y *Usuario* (el tipo de frases utilizadas por los usuarios, siendo en su mayoría de tipo interrogativo).

- **CASTELLANO:** en este campo es donde se insertan las oraciones en castellano, que son las seleccionadas previamente. Estas frases se encuentran representadas en minúsculas.

- **LSE:** aquí se insertan las frases de la Lengua de Signos Española, representadas por glosas. Las glosas son palabras (en mayúsculas) que representan los signos. Por ejemplo la glosa FOTO representa el signo cuyo significado es el de “fotografía”.Éstas son las frases que han sido traducidas por personas sordas, cuyo equivalente se encuentra en la misma fila, en *CASTELLANO*. Para diferenciarlas, las glosas están representadas en mayúsculas.

- **VIDEO:** en este campo se indican enlaces a archivos de video (tanto archivos de tipo *avi* como archivos de tipo *wmv*). En estos videos se representa la frase completa resultado de la traducción. Los videos han sido grabados por las personas sordas que se han encargado de la traducción. Han sido representadas la mayoría de las frases, para ayudar a generar los signos.



	A	B	C	D	E
9	Policía	por favor espere su turno en la cola	POR-FAVOR TU FILA ESPERAR	videos2\por favor espere su turno en la cola	
10	Policía	va a tener usted que volver a hacerse las fotos	TU FOTO OTRA-VEZ DEBER	videos2\va a tener usted que volver a hacers	
11	Policía	puede usted recogerlo aquí mismo	TU DNI AQUI COGER PODER	videos2\puede usted recogerlo aqui mismo	
12	Usuario	¿qué hay que hacer para sacarse el dnei?	DNI EMPEZAR PAPELES NECESITAR CUAL?	videos2\¿que hay que hacer para sacarse e	
13	Usuario	¿durante cuánto tiempo es válido el dnei?	DNI PLAZO CUANDO?	videos2\¿durante cuanto tiempo hay que re	
14	Usuario	¿cada cuanto tiempo hay que renovar el dnei?	DNI RENOVAR CUANDO?	videos2\¿cada cuanto tiempo hay que renov	
15	Usuario	¿qué papeles hacen falta para sacarse el dnei?	DNI RENOVAR PAPELES NECESITAR CUAL?	videos2\¿que papeles hace falta para sacar	
16	Usuario	¿hace falta estar empadronado?	CERTIFICADO PADRON NECESITAR?	videos2\¿hace falta estar empadronado.avi	
17	Usuario	¿cuánto tardan en facilitarme el dnei nuevo?	DNI NUEVO RECOGER CUANDO?	videos2\¿cuanto tarda en facilitarme el dni n	
18	Usuario	¿es aquí donde renuevan el dnei?	AQUI DNI RENOVAR?	videos2\¿es aqui donde renueva el dni.avi	
19	Usuario	¿dónde debo dirigirme para renovar el dnei?	DNI RENOVAR DONDE?	videos2\¿donde debo dirigirme para renova	
20	Usuario	¿cuál es el horario de atención?	HORARIO ATENCION ABRIR CERRAR HORA?	videos2\¿cuál es el horario de atención.wm	
21	Usuario	¿cuánto cuesta renovar el dnei?	DNI RENOVAR CUANTO?	videos2\¿cuanto cuesta renovar el dni.avi	
22	Generales	adiós buenos días	ADIOS	videos2\adios buenos tardes.wmv	
23	Generales	adiós buenas tardes	ADIOS	videos2\adios buenos dias.wmv	
24	Policía	te lo tienes que sacar a los catorce años obligato	EDAD CATORCE DNI OBLIGATORIO	videos2\te lo tienes que sacar a los catorce	
25	Policía	ven conmigo por favor	POR-FAVOR CONMIGO	videos2\VEN CONMIGO POR FAVOR.wmv	
26	Policía	ven conmigo	VEN CONMIGO	videos2\ven conmigo.avi	
27	Policía	debes rellenar este impreso	PAPEL ESTE TU ESCRIBIR DEBER	videos2\debes rellenar este impreso.wmv	
28	Policía	necesitas rellenar este impreso	PAPEL ESTE TU ESCRIBIR DEBER	videos2\necesitas rellenar este impreso.wm	
29	Policía	debes tener dos fotografías	FOTOS DOS DEBER	videos2\debes tener dos fotografias.avi	
30	Policía	necesitas dos fotografías	FOTOS DOS DEBER	videos2\necesitas dos fotografias.avi	
31	Policía	necesita dos fotos	FOTOS DOS ACTUAL DEBER	videos2\necesita dos fotografias.avi	
32	Policía	necesitas dos fotos	FOTOS DOS NECESITAR	videos2\necsitas dos fotos.avi	
33	Policía	son veinte euros con sesenta	PRECIO VEINTE COMA SESENTA EUROS	videos2\son veinte euros con sesenta.avi	
34	Usuario	lo he perdido	YO DNI PERDER	videos2\lo he perdido.avi	
35	Policía	en el caso de pérdida son cuarenta euros con ve	ESPECIFICO DNI PERDER PRECIO CUARENTA COMA VEINTE EUROS	videos2\en caso de perdida son cuarenta e	
36	Policía	¿cómo te llamas?	TU NOMBRE?	videos2\¿como te llamas.wmv	
37	Policía	dime tu nombre	TU NOMBRE DECIR-A_MI	videos2\dime tu nombre.avi	
38	Policía	¿cómo te llamas?	TU NOMBRE?	videos2\¿como te llamas.wmv	
39	Policía	dime tu dirección	TU CALLE NOMBRE DECIR-A_MI	videos2\dime tu direccion.avi	
40	Usuario	tengo que renovar el dnei	YO DNI RENOVAR DEBER	videos2\tengo que renovar el dni.avi	

Originales | Ampliación por el GTH

Figura 14: Aspecto del fichero de tipo *Excel* de la base de datos de ejemplos para el DNI

El número de frases que encontramos en cada una de las tablas, diferenciando el tipo de cada una de ellas, es el siguiente:

- **Tabla *Originales*:** contiene un total de 301 frases.
  - *Generales*: contiene 21 frases
  - *Funcionario*: contiene 229 frases
  - *Usuario*: contiene 51 frases
- **Tabla *Ampliación por el GTH*:** contiene un total de 322 frases.
  - *Generales*: contiene 8 frases
  - *Funcionario*: contiene 217 frases
  - *Usuario*: contiene 97 frases

3.2. Base de datos para la renovación del carné de conducir

Para este dominio de aplicación solicitamos la colaboración de la DGT, concretamente de la Jefatura Provincial de Tráfico de Toledo. Realizamos un viaje a esta oficina y les solicitamos ayuda para recoger las frases que con mayor frecuencia pronuncian los funcionarios en su atención personal.

Figura 15: Entrada de la oficina de la JPT de Toledo

Figura 16: Diferentes ventanillas de la oficina

Esta oficina está organizada en 5 ventanillas principales:

- La ventanilla de información que te da información general y te redirige a una u otra ventanilla.
- La ventanilla de caja donde se abonan las tasas correspondientes.
- La ventanilla conductores donde se realizan las gestiones relacionadas con los conductores: renovación del carné, duplicados,…
- La ventanilla vehículos donde se realizan las gestiones relacionadas con los vehículos: alta, baja,….
- La ventanilla de autoescuelas

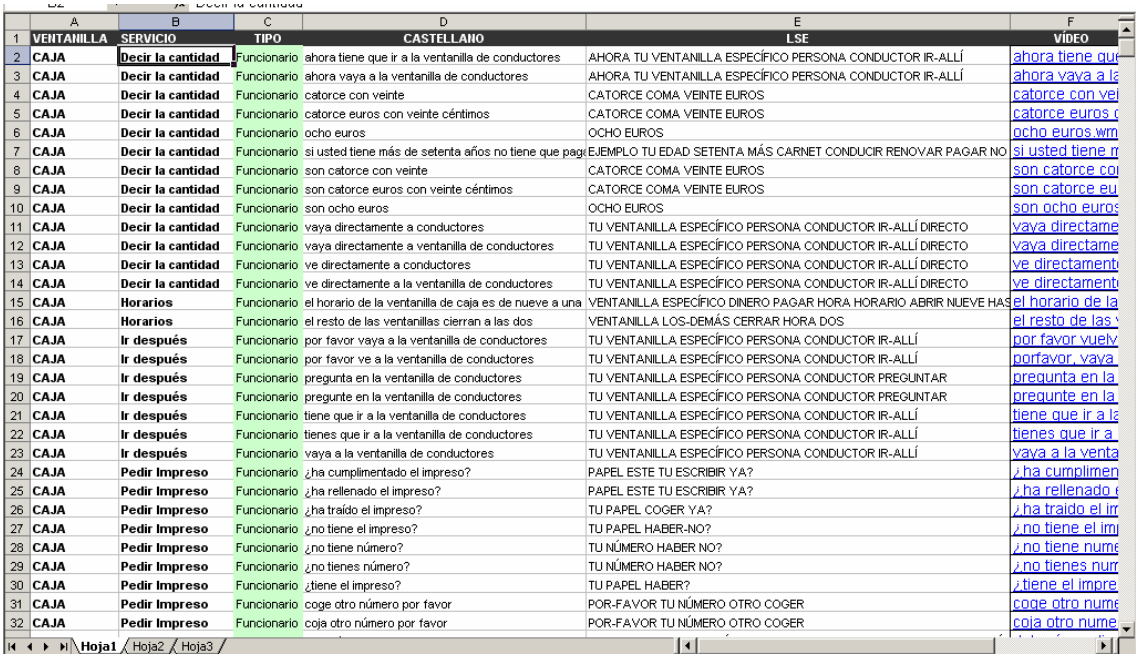
Se solicitó la ayuda de los funcionarios para recoger las frases que con mayo frecuencia pronunciaban cuando atendían a las diferentes personas. La respuesta de estos funcionarios y de la oficina en general fue excepcional generando más de 4000 frases en tres semanas: alrededor de 1000 frases por cada una de las ventanillas.

Esta cantidad de frases desbordaron nuestras expectativas iniciales por lo que hubo que realizar una selección de dichas frases para realizar el estudio lingüístico y traducirlas a LSE. Esta selección se realizó acotando el servicio ofrecido. En nuestro caso en particular, se seleccionaron las frases

relacionadas con la renovación del carné de conducir. Este servicio implicó la selección de las frases de varias ventanillas: información, caja y conductores. Esta selección ha sido posible gracias a que todas las frases recogidas están etiquetas con la ventanilla y el servicio ofrecido dentro de esta ventanilla. El proceso de generación de la base de datos ha sido equivalente al de la base de datos anterior.

Esta base de datos se encuentra en un fichero de tipo Excel (igual que el caso anterior). El fichero en el que se guardan todas las frases es: frases_DGT.xls. Dentro de este fichero de tipo Excel encontramos una única hoja con una tabla compuesta por:

- **VENTANILLA:** ventanilla en la que se recogieron las frases.
- **SERVICIO:** servicio que se estaba ofreciendo en esa ventanilla.
- **TIPO:** en este campo se indica la procedencia de las frases, pudiendo diferenciar tres tipos: *Generales* (son el tipo de frases utilizadas tanto por el usuario como por el funcionario), *Funcionario* (frases utilizadas por el funcionario de la Administración) y *Usuario* (el tipo de frases utilizadas por los usuarios, siendo en su mayoría de tipo interrogativo).
- **CASTELLANO:** en este campo es donde se insertan las oraciones en castellano, que son las seleccionadas previamente. Estas frases se encuentran representadas en minúsculas.
- **LSE:** aquí se insertan las frases de la Lengua de Signos Española, representadas por glosas. Éstas son las frases que han sido traducidas por personas sordas, cuyo equivalente se encuentra en la misma fila, en *CASTELLANO*. Para diferenciarlas, las glosas están representadas en mayúsculas.
- **VIDEO:** en este campo se indican enlaces a archivos de video (tanto archivos de tipo *avi* como archivos de tipo *wmv*). En estos videos se representa la frase completa resultado de la traducción. Los videos han sido grabados por las personas sordas que se han encargado de la traducción. Han sido representadas la mayoría de las frases, para ayudar a generar los signos.



	A	B	C	D	E	F
1	VENTANILLA	SERVICIO	TIPO	CASTELLANO	LSE	VÍDEO
2	CAJA	Decir la cantidad	Funcionario	ahora tiene que ir a la ventanilla de conductores	AHORA TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR IR-ALLÍ	ahora tiene qu
3	CAJA	Decir la cantidad	Funcionario	ahora vaya a la ventanilla de conductores	AHORA TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR IR-ALLÍ	ahora vaya a la
4	CAJA	Decir la cantidad	Funcionario	catorce con veinte	CATORCE COMA VEINTE EUROS	catorce con vei
5	CAJA	Decir la cantidad	Funcionario	catorce euros con veinte céntimos	CATORCE COMA VEINTE EUROS	catorce euros c
6	CAJA	Decir la cantidad	Funcionario	ocho euros	OCHO EUROS	ocho euros.wm
7	CAJA	Decir la cantidad	Funcionario	si usted tiene más de setenta años no tiene que pag	EJEMPLO TU EDAD SETENTA MÁS CARNET CONDUCIR RENOVAR PAGAR NO	si usted tiene m
8	CAJA	Decir la cantidad	Funcionario	son catorce con veinte	CATORCE COMA VEINTE EUROS	son catorce co
9	CAJA	Decir la cantidad	Funcionario	son catorce euros con veinte céntimos	CATORCE COMA VEINTE EUROS	son catorce eu
10	CAJA	Decir la cantidad	Funcionario	son ocho euros	OCHO EUROS	son ocho euros
11	CAJA	Decir la cantidad	Funcionario	vaya directamente a conductores	TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR IR-ALLÍ DIRECTO	vaya directame
12	CAJA	Decir la cantidad	Funcionario	vaya directamente a ventanilla de conductores	TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR IR-ALLÍ DIRECTO	vaya directame
13	CAJA	Decir la cantidad	Funcionario	ve directamente a conductores	TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR IR-ALLÍ DIRECTO	ve directamente
14	CAJA	Decir la cantidad	Funcionario	ve directamente a la ventanilla de conductores	TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR IR-ALLÍ DIRECTO	ve directamente
15	CAJA	Horarios	Funcionario	el horario de la ventanilla de caja es de nueve a una	VENTANILLA ESPECÍFICO DINERO PAGAR HORA HORARIO ABRIR NUEVE HAS	el horario de la
16	CAJA	Horarios	Funcionario	el resto de las ventanillas cierran a las dos	VENTANILLA LOS-DEMÁS CERRAR HORA DOS	el resto de las
17	CAJA	Ir después	Funcionario	por favor vaya a la ventanilla de conductores	TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR IR-ALLÍ	por favor vuelv
18	CAJA	Ir después	Funcionario	por favor ve a la ventanilla de conductores	TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR IR-ALLÍ	porfavor, vaya
19	CAJA	Ir después	Funcionario	pregunta en la ventanilla de conductores	TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR PREGUNTAR	pregunta en la
20	CAJA	Ir después	Funcionario	pregunte en la ventanilla de conductores	TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR PREGUNTAR	pregunte en la
21	CAJA	Ir después	Funcionario	tiene que ir a la ventanilla de conductores	TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR IR-ALLÍ	tiene que ir a la
22	CAJA	Ir después	Funcionario	tienes que ir a la ventanilla de conductores	TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR IR-ALLÍ	tienes que ir a
23	CAJA	Ir después	Funcionario	vaya a la ventanilla de conductores	TU VENTANILLA ESPECÍFICO PERSONA CONDUCTOR IR-ALLÍ	vaya a la venta
24	CAJA	Pedir Impreso	Funcionario	¿ha cumplimentado el impreso?	PAPEL ESTE TU ESCRIBIR YA?	¿ha cumplimen
25	CAJA	Pedir Impreso	Funcionario	¿ha rellenado el impreso?	PAPEL ESTE TU ESCRIBIR YA?	¿ha rellenado e
26	CAJA	Pedir Impreso	Funcionario	¿ha traído el impreso?	TU PAPEL COGER YA?	¿ha traído el im
27	CAJA	Pedir Impreso	Funcionario	¿no tiene el impreso?	TU PAPEL HABER-NO?	¿no tiene el imp
28	CAJA	Pedir Impreso	Funcionario	¿no tiene número?	TU NÚMERO HABER NO?	¿no tiene nume
29	CAJA	Pedir Impreso	Funcionario	¿no tienes número?	TU NÚMERO HABER NO?	¿no tienes num
30	CAJA	Pedir Impreso	Funcionario	¿tiene el impreso?	TU PAPEL HABER?	¿tiene el impre
31	CAJA	Pedir Impreso	Funcionario	coge otro número por favor	POR-FAVOR TU NÚMERO OTRO COGER	coge otro nume
32	CAJA	Pedir Impreso	Funcionario	coja otro número por favor	POR-FAVOR TU NÚMERO OTRO COGER	coja otro nume

Hoja1 / Hoja2 / Hoja3

Figura 17: Aspecto del fichero de tipo *Excel* de la base de datos de ejemplos para la DGT

El número de frases que encontramos es de 668:

- *Generales*: contiene 31 frases
- *Funcionario*: contiene 507 frases
- *Usuario*: contiene 130 frases

3.3. Reglas para el etiquetado de frases en LSE mediante glosas

A continuación vamos a describir las principales reglas de etiquetado utilizadas en la generación de las frases para la traducción de la base de datos. Las reglas han sido generadas en colaboración con la Fundación CNSE pensando en el desarrollo de sistemas automáticos de traducción.

1) Las glosas van especificadas con palabras en mayúsculas:
   - Ejemplo: DNI EDAD EMPEZAR CUANDO?.
2) Cuando varias glosas se unen para formar la descripción de un signo, no debe haber un espacio en blanco. Los espacios en blanco sirven para diferenciar las glosas correspondientes a signos diferentes.
3) Cada glosa o conjunto de glosas debe corresponder unívocamente a un único signo. Esta correspondencia unívoca supone:
   - Que glosas diferentes no deben dar lugar al mismo signo:
     - Ejemplo: si tenemos las glosas FOTO, FOTOS, FOTOGRAFÍA y hacen referencia al mismo signo (“foto”), todas las glosas se deberían unificar.

- Que una misma glosa o conjunto de glosas no dé lugar a signos diferentes según el contexto. En ese caso sería necesario generar tantas glosas (o conjuntos de glosas) como signos diferentes haya.

4) El carácter "+" significa que las dos glosas componen un signo compuesto
   - Ejemplo: PAPA+ MAMA → padres.
5) El carácter "-" significa que esas glosas, aunque en castellano correspondan con diferentes palabras, en LSE son un único signo:
   - Ejemplo: TODO-EL-DIA → es un único signo que significa "todo el día".
   - Ejemplo: ENCONTRAR-NO si son dos signos diferentes debemos representarlos como glosas separadas ENCONTRAR NO.
6) Los caracteres "dl" (ambos en minúsculas) sirven para indicar que la glosa que viene después se deletrea (dactilográficamente):
   - Ejemplo: dlRUBEN debe generar la secuencia de signos R U B E N (cada letra se representa con un signo diferente).
   - Si tenemos varios nombres seguidos, debemos poner "dl" delante de todas las glosas que se vayan a deletrear. Como por ejemplo en el caso siguiente: dlRUBEN dlSANSEGUNDO.
7) Cuando al final de un conjunto de glosas que forman un signo aparecen varios caracteres "+" indica que el signo correspondiente al conjunto de glosas se debe realizar tantas veces como caracteres "+" se pongan:
   - Ejemplo: TRABAJAR++ es equivalente a TRABAJAR TRABAJAR (se representaría dos veces el signo TRABAJAR).
8) La partícula CL quiere decir que son clasificadores. Un clasificador pretende describir un concepto, situación, movimiento…, para el que no existe una glosa específica. Existen varios tipos de clasificadores como: descriptivo, instrumental… Se añade con CLD, CLI… Seguido a las siglas del clasificador es necesario incluir una descripción entre comillas del concepto a representar, usando palabras en minúsculas. Esta descripción no debe contener espacios ni comas: se deben conectar las palabras con guiones:
   - Ejemplo: CLL"debe-colocarse-en-el-centro-no-al-lado". La descripción tiene que ser lo suficientemente amplia como para saber cómo representar el signo. Un mal ejemplo sería CD"dedo" sería mejor CD"limpiar-dedo".
9) Los signos de puntuación no los consideraremos en el proceso de etiquetado. El único signo que utilizaremos será la "," para reflejar una pausa.

10) En algunas situaciones puede ser necesario definir un nuevo signo, como por ejemplo FOTOMATON o DNI-ELECTRONICO. En este caso, aunque no tengamos muy claro la representación del signo, sí podemos definir una glosa sencilla. Esta glosa puede convertirse (al generar los signos en HamNoSys) en una paráfrasis inicialmente, pero luego, a medida que se vaya consensuando el signo, cambiar la descripción del signo en HamNoSys sin necesidad de cambiar la glosa.

11) La representación de los números la vamos a realizar siempre con glosas en letras. Consideraremos una única glosa para cada uno de los números del 0 al 99:

    - Ejemplo: UNO para el “1”, DOS para el “2”,…., VEINTITRÉS para el “23”, CUARENTAYUNO para el “41”.
    - En el caso de números con decimales pondremos la glosa COMA para diferenciar los decimales:
        - Ejemplo: 40,13 en glosas se etiquetaría: CUARENTA COMA TRECE.

12) En relación con las horas, éstas se representan de la siguiente forma:

    - Ejemplo 13:15 en glosas sería: HORA TRECE Y-CUARTO.
    - Ejemplo: 15:45 en glosas sería: HORA DIECISEIS MENOS-CUARTO.
    - Ejemplo: 8:30 en glosas seria: HORA OCHO Y-MEDIA.
    - También es posible etiquetar horas exactas:
        - Ejemplo: 12:35 → HORA DOCE TREINTAYCINCO.
        - En estos casos no debe incluirse la glosa Y.

13) Para el etiquetado de las fechas se sigue la siguiente regla:

    - Ejemplo: 4-abril-2008 en glosas sería: DIA CUATRO ABRIL DOS MIL OCHO, o dando por sobrentendido el año: DIA CUATRO ABRIL.
    - A veces se puede omitir la glosa DIA porque queda claro que es una fecha, como en el caso siguiente: 21-abril en glosas sería VEINTIUNO ABRIL.

14) En el caso en el que haya varias posibilidades de signar lo mismo con el mismo significado conviene elegir una de las opciones y ser coherente a lo largo de todo el etiquetado:

    - Ejemplo: PERDER y ENCONTRAR NO. Se debe elegir el conjunto de glosas/signos que mejor se adapte y utilizarlo siempre así.

4. Arquitectura del sistema de traducción de voz a LSE

La arquitectura software del sistema de traducción de voz a LSE se muestra en la siguiente figura.

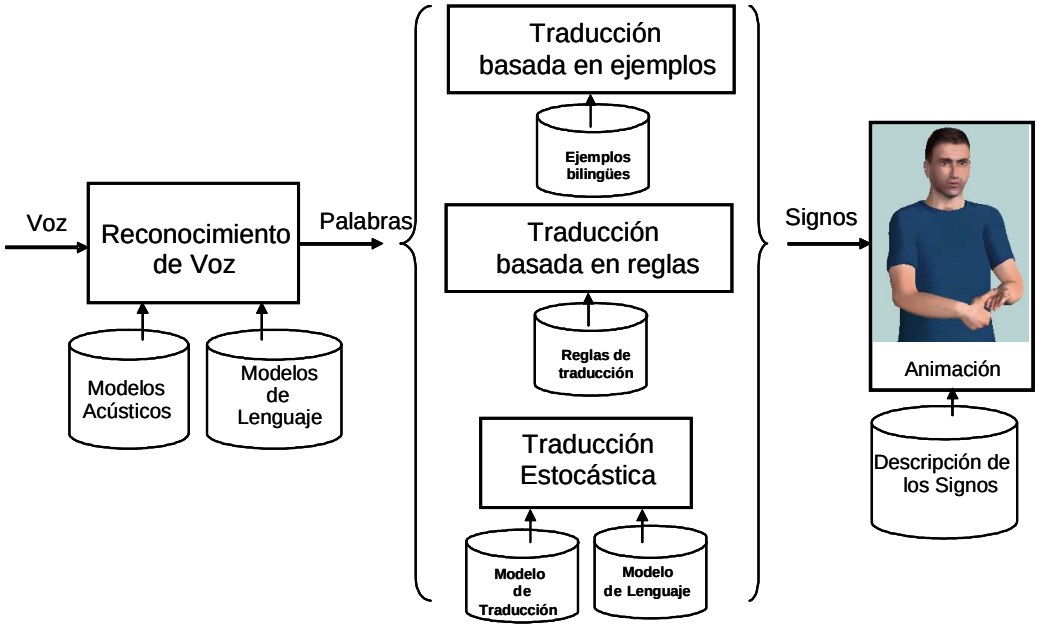


Figura 18. Etapas del sistema a desarrollar.

A grandes rasgos, el proceso de traducción se realizará en 3 pasos (Figura 18):

- En el primer paso, un reconocedor de voz que se encarga de traducir la voz en una secuencia de palabras. Este módulo permite reconocer habla en lenguaje natural (habla continua) e independiente del locutor.

- En segundo lugar, un módulo de traducción automática traduce la secuencia de palabras en una secuencia de signos. Para la implementación de este módulo se han desarrollado tres alternativas tecnológicas. La primera se basa en una base de datos de ejemplos, de forma que se propone como resultado de la traducción, la traducción del ejemplo de la base de datos que más se parece a la frase de entrada. La segunda estrategia se basa en reglas de traducción desarrolladas por una persona experta. Y por último, la tercera estrategia está basada en métodos estadísticos cuyos modelos se aprenden a partir de un corpus paralelo frases de texto-secuencias de signos. La combinación de las diferentes estrategias de traducción se describirá en detalle en el capítulo 6 dedicado a la traducción de texto a LSE.

- Finalmente, el módulo de animación de los signos basado en un agente animado en 3D (incorporado en el sistema como un control ActiveX). Este agente animado ha sido desarrollado en el proyecto eSIGN (http://www.sign-lang.uni-hamburg.de/eSIGN/) y está disponible para investigación.

5. Reconocimiento automático de voz

El reconocimiento de voz es el proceso automático de conversión a texto de las frases pronunciadas en el lenguaje natural. Tras un proceso de parametrización y extracción de características de la voz, se procede al reconocimiento como tal. Cualquier reconocedor de voz está formado por tres elementos fundamentales: el modelo acústico, el modelo de lenguaje y el vocabulario empleado.

- Modelo acústico. Proporciona información sobre las propiedades y las características de los sonidos, permitiendo la identificación de los mismos.
- Modelo de lenguaje. Complementa el conocimiento acústico con información sobre las secuencias más probables de palabras. Contiene información de cómo se deben combinar las palabras para formar frases.
- Vocabulario. Diccionario de palabras que pueden ser reconocidas por el sistema.

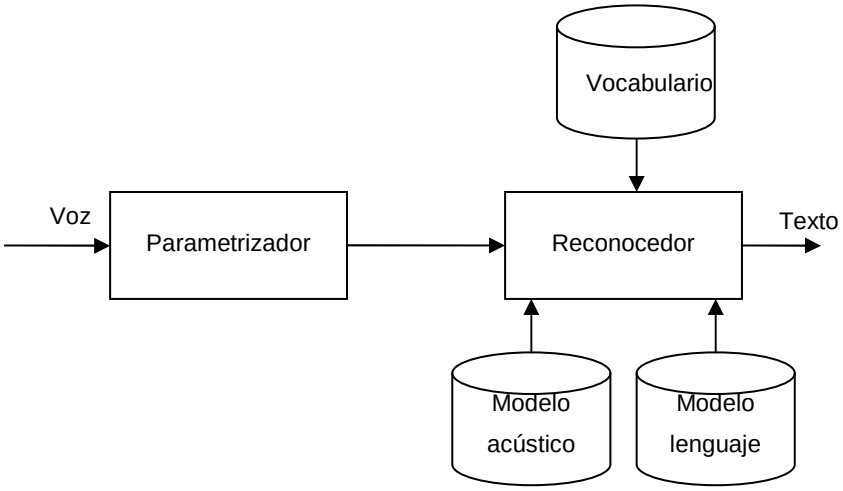


Figura 19. Esquema de un reconocedor de voz

Para la elaboración del módulo de reconocimiento de voz se ha utilizado la plataforma para el desarrollo de aplicaciones con voz desarrollada por el Grupo de Tecnología del Habla (GTH-UPM) y denominada SERVIVOX. Es un sistema basado en modelos ocultos de Markov1, con las siguientes características:

[1] **Modelos ocultos de Markov.** Los modelos ocultos de Markov son modelos matemáticos basados en probabilidades que pueden ser adaptados para resolver problemas de reconocimiento de voz. Es un modelo capaz de describir hechos acústicos del habla y que queda completamente definido por medio de una serie de variables estadísticas.

- Es un sistema de reconocimiento de habla continua, es decir, reconoce varias palabras habladas de forma continua.

- Independencia del locutor. El reconocedor ha sido entrenado con una base de datos grande, conteniendo más de 20 horas de habla de 4000 locutores distintos. Se dota así al sistema de una gran robustez frente al amplio rango de locutores potenciales, por lo que no es necesario realizar un entrenamiento adicional para cada nuevo usuario del sistema.

- El reconocedor proporciona un valor de medida de confianza, entre cero (confianza más baja) y 1 (confianza más alta), para cada palabra reconocida. Esta medida es muy importante, ya que el reconocedor puede verse afectado por aspectos como el nivel de ruido ambiente, locutores no nativos, habla más o menos espontánea, o similitud acústica entre palabras distintas del vocabulario.

- Ofrece la posibilidad de adaptar los modelos acústicos a un entorno acústico o a un grupo de locutores, mejorando sensiblemente la tasa de reconocimiento y aumentando la velocidad de proceso.

Respecto al modelo acústico, el reconocedor utiliza 5760 modelos ocultos de Markov para modelar todos los posibles alófonos y su contexto: diferencia entre sonidos fuertes, débiles o nasales, incluyendo también diferentes variantes para el sonido vibrante de la “r”, diptongos, la versión fricativa de “b”, “d” y “g”, y las africadas de “y”. El sistema también tiene 16 modelos de ruido y silencio, para detectar efectos acústicos que aparecen en el habla espontánea, como ruido de fondo, ruidos del locutor (carraspeos…), pausas, etc. Es importante detectar y procesar estos efectos para evitar que afecten al reconocedor.

Respecto al modelo de lenguaje, debido a la escasez de los datos, en este sistema se emplea un modelo bigrama, ya que hay un número pequeño de frases para entrenar el modelo en comparación con el tamaño del vocabulario, y por lo tanto, en comparación con la cantidad de n-gramas diferentes.

Se ha realizado una evaluación interna del reconocedor con tres locutores diferentes, grabando más de 500 frases por cada locutor. El reconocedor a ofrecido una tasa del 92,5% de acierto de palabra (WER: Word Error Rate) en estos dominios de aplicación.

6. Traducción de castellano a LSE

A continuación se describen las tres soluciones tecnológicas implementadas para la traducción de texto en castellano a LSE. Finalmente se describe la estrategia de combinación de dichas soluciones.

6.1. Traducción basada en ejemplos

Este tipo de traducción se basa en la comparación de la frase a traducir con varias frases (ejemplos) de una base de datos y sus correspondientes traducciones. La traducción del ejemplo al que más se parezca la secuencia de palabras reconocidas será el resultado de la traducción. Dentro de esta comparación es muy importante la categorización (asignar a ciertas palabras una categoría genérica) de los ejemplos, así como de la frase a traducir, ya que se trata de comparar ciertos patrones.

A continuación vamos a describir un ejemplo concreto de traducción basada en ejemplos dentro de nuestro sistema. Supongamos que debemos traducir la siguiente secuencia de palabras: “mi cita es para las cinco y diez”. Lo primero que debemos hacer es categorizar la frase. En la frase hay dos palabras a los que podemos asignarles una categoría, “cinco” y “diez”. Las dos palabras habría que etiquetarlas como números (categoría $NUMERO). A continuación debemos comparar esta frase categorizada con los ejemplos de la base de datos. El ejemplo más parecido es: “mi cita es para las $NUMERO y $NUMERO” cuya traducción es en glosas YO CITA-PREVIA HORA $NUMERO $NUMERO. Sólo queda por tratar las palabras categorizadas, que si las traducimos a glosas (debido a que los signos están etiquetados mediante las glosas, para traducir únicamente debemos convertir la palabra a mayúsculas) quedaría CINCO y DIEZ, respectivamente. Por tanto, el resultado de la traducción sería: YO CITA-PREVIA HORA CINCO DIEZ.

En la siguiente figura vemos como el sistema de traducción es capaz de traducir incluso cuando hay algún error de reconocimiento. En este caso no se ha reconocido bien el artículo “el”.

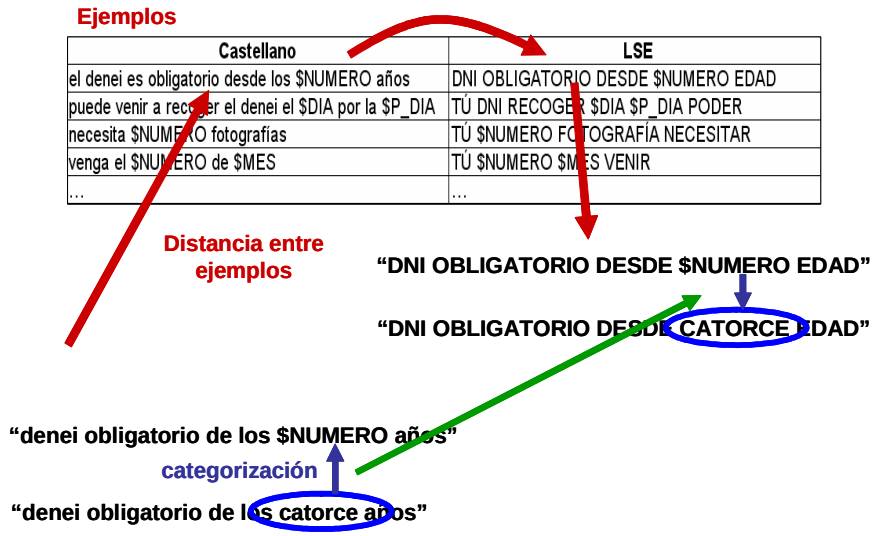


Figura 20: Ejemplo de traducción basada en ejemplos

La parte fundamental del método de traducción basada en ejemplos es la comparación con los ejemplos de la base de datos. Para comparar la frase a traducir con los ejemplos de la base de datos utilizamos como medida la distancia de Levenshtein. Mediante esta medida podemos comprobar cuál es el ejemplo más cercano a la frase a traducir, obteniendo de esta manera el resultado de la traducción.

La distancia de Levenshtein es una métrica de medida de comparación entre dos secuencias. Se suele utilizar para medir la diferencia entre dos secuencias de caracteres, aunque en nuestro caso en lugar de considerar caracteres utilizamos palabras. La distancia de Levenshtein entre dos secuencias de caracteres es el número mínimo de operaciones de edición (inserción de un carácter, borrado de un carácter o sustitución de un carácter por otro) a realizar para convertir una secuencia en la otra. Se puede decir que es una generalización de la distancia de Hamming, aunque la distancia de Levenshtein admite secuencias de distinta longitud. Así, la distancia de Levenshtein entre las secuencias “silla” y “tina” es de 3 (3 operaciones de edición). Existen varias formas de realizar estas 3 operaciones, siendo una de ellas:

- **t**ina->**s**ina (sustitución de ‘t’ por ‘s’)
- sina->si**l**na (inserción de ‘l’ entre ‘i’ y ‘n’)
- sil**n**a->sil**l**a (sustitución de ‘n’ por ‘l’)

Para calcular la distancia de Levenshtein se puede utilizar un algoritmo de programación dinámica basado en el algoritmo Wagner-Fisher de distancia de edición, en el que se va rellenando de abajo a arriba una matriz de (n+1)x(m+1), siendo n y m el número de elementos de cada una de las secuencias.

6.1.1. Algoritmo utilizado para calcular la distancia de Levenshtein

Supongamos que tenemos dos secuencias de enteros, s[1..m] y t[1..n]. La matriz que vamos a rellenar la denominaremos d[0..m, 0..n]:

- Para cada **i** de 0 a m:
    - d[i, 0] = i;
- Para cada **j** de 0 a n:
    - d[0, j] = j;
- Para cada **i** de 0 a m:
- Para cada **j** de 0 a n:
    - Si s[i] == t[j]:

- coste = 0;

- Si s[i] != t[j]:
  - coste = SUB_COST;
- d[i, j] = min {d[i-1, j] + DEL_COST, d[i, j-1] + INS_COST, d[i-1, j-1] + coste};

- Devolvemos el valor d[m, n];

Tabla 1: Algoritmo utilizado para el cálculo de la distancia de Levenshtein

Los pesos de cada una de las operaciones utilizados son:

- Inserción→INS_COST = 1.
- Borrado→DEL_COST = 1.
- Sustitución→SUB_COST = 1. En nuestro caso le hemos asignado un peso de 1, aunque en la literatura existen casos en los que se le asigna el valor de 2, ya que una sustitución implica un borrado y una posterior inserción.

El algoritmo que utilizamos queda implementado en la función *Distancia*, que se encuentra en el fichero *frases_lin_reglas.cpp*. Esta función calcula la distancia de Levenshtein mediante el algoritmo comentado, aunque además calcula el camino de operaciones seguido en la matriz, que se va guardando en **PathMatrix**. Este cálculo sirve para obtener al final el tipo de operaciones realizadas durante el camino de menor distancia, aunque en la función *Distancia* no se devuelve.

En la siguiente tabla podemos ver el cálculo de la distancia de Levenshtein para el caso del ejemplo anterior, entre las secuencias “silla” y “tina”. En negrita se ha señalado el camino seguido (en diagonal las sustituciones y hacia la derecha las inserciones):

		S	**I**	**L**	**L**	**A**
	0	1	2	3	4	5
T	1	**1**	2	3	4	5
I	2	2	**1**	**2**	3	4
N	3	3	2	2	**3**	4
A	4	4	3	3	3	**3**

Tabla 2: Distancia de Levenshtein entre “silla” y “tina”

6.1.2. Desarrollo del método de traducción

Dentro del sistema de traducción, como ya hemos comentado, la función que desarrolla todo el método de traducción es *traduceLineal()*. Esta función se encarga de realizar la traducción, rellenado la variable **ComprensionLineal**.

Esta variable se inicializa en el arranque de la aplicación, dentro de la función *OnInitDialog()* de la clase principal *LsetovozDlg.cpp*. Para inicializar **ComprensionLineal** se realizan tres operaciones básicas:

- Reservar memoria
- Cargar los ejemplos de la base de datos
- Cargar las categorías de los signos

Para reservar memoria para la variable utilizamos la función *ReservaMemoriaComprension*, a la que le pasamos la dirección de **ComprensionLineal**. Esta función se encuentra en la clase *primitivas.cpp*.

La función *LeerParametros* es la encargada de cargar los ejemplos de la base de datos. Los parámetros que le pasamos a esta función son los nombres de los ficheros donde se encuentran los ejemplos. Los ficheros donde se encuentran los ejemplos de la base de datos son cuatro:

- **PATH_SIGNOS_LINEAL:** se encuentra en .*TRADUCCION**LINEAL**signos.txt*. En cada línea del archivo de texto se encuentra un ejemplo, representando la secuencia de signos representados en glosas correspondiente.
- **PATH_FRASES_LINEAL:** se encuentra en .*TRADUCCION**LINEAL**frases.txt*. En cada línea del archivo de texto se encuentra un ejemplo (correspondiente a la traducción en castellano del ejemplo de la misma línea en *signos.txt*).
- **PATH_SIGNOS_CAT_LINEAL:** este archivo se encuentra en la siguiente ruta: .*TRADUCCION**LINEAL**signos_cat.txt*. El archivo es el mismo que *signos.txt*, con la diferencia que en este caso se encuentra categorizado.
- **PATH_FRASES_CAT_LINEAL:** este archivo se encuentra en la siguiente ruta: .*TRADUCCION**LINEAL**frases_cat.txt*. Es el mismo archivo que *frases.txt*, estando en este caso categorizadas las palabras.

Una vez leídos estos cuatro ficheros, dentro de *LeerParametros* se cargan los ejemplos, utilizando para ello una variable de tipo *struct* ***param***, denominada en este caso **lista**. La función *LeeParametros* devolverá una variable de este tipo con todos los ejemplos cargados, de los cuatro archivos. Dentro de *OnInitDialog()* es donde recogemos la variable devuelta por la función en la variable **lista**. La función se encuentra definida en el archivo

param.cpp. La definición del tipo *struct* ***param*** se encuentra en el archivo *param.h*, junto con la definición de otras dos variables externas que se utilizarán dentro de *LeerParametros*, como son **lista_pal** y **num_palabras**. Las operaciones básicas que se realizan dentro de *LeerParametros* para cargar los ejemplos de la base de datos son:

- Se cargan en **lista_pal** todas las palabras de los cuatro ficheros (tanto palabras en castellano como las glosas de los signos), sin repetir.
- Para cada ejemplo (línea de los ficheros):
  - Cargamos la línea correspondiente de cada fichero:
    - **lista->frase_orig:** línea de *frases.txt*.
    - **lista->frase_orig_cat:** línea de *frases_cat.txt*.
    - **lista->frase_dest:** línea de *signos.txt*.
    - **lista->frase_dest_cat:** línea de *signos_cat.txt*.
  - Cargamos cada palabra de la línea correspondiente de cada fichero, mediante su índice correspondiente de **lista_pal**:
    - **lista->p_e:** índices de la línea de *frases.txt*.
    - **lista->p_e_cat:** índices de la línea de *frases_cat.txt*.
    - **lista->p_s:** índices de la línea de *signos.txt*.
    - **lista->p_s_cat:** índices de la línea de *signos_cat.txt*.
  - Pasamos al siguiente ejemplo (**lista->next**).

Tabla 3: Esquema de *LeerParametros(…)*

Cada ejemplo de la base de datos lo vamos cargando en **lista**, mediante el parámetro **next**. Este parámetro apunta a otra variable de tipo *struct* ***param***, que es donde iremos cargando el siguiente ejemplo. El parámetro **next** del último de los ejemplos será NULL.

Una vez cargados los ejemplos de la base de datos, queda cargar el fichero de categorías, donde se encuentran todas las palabras de los ejemplos, indicando en cada una de ellas su categoría. La mayoría de las palabras no están categorizadas, por lo que la categoría asignada es la propia palabra. Las palabras que sí se encuentren categorizadas tendrán asignada una de estas categorías:

- **$NUMERO:** las glosas correspondientes a los números del 0 (cero) al 100 (cien).
- **$MES:** las glosas correspondientes a los 12 meses (enero, febrero,…).

- **$DIA_SEMANA:** las glosas correspondientes a los 7 días de la semana (lunes, martes,…).

- **$DELETREO:** las glosas deletreadas letra a letra, principalmente nombres propios.

- **$ORDINAL:** las glosas correspondientes a los números ordinales, del 1 al 10, y las correspondientes a las decenas hasta el 100 (primero, segundo,…).

- **$LETRA:** las glosas correspondientes a las 27 letras del abecedario.

Estas categorías son las indicadas también en el fichero *frases_cat.txt*, sustituyendo a las palabras correspondientes. En el fichero *signos_cat.txt* también se han indicado estas categorías, sustituyendo en este caso a la palabra en castellano equivalente.

El fichero donde se indican todas las glosas de los ejemplos, indicando su categoría, se encuentra en la siguiente ruta: *.\TRADUCCION\LINEAL\categorias_lineal.txt*. Este fichero también lo cargamos en *OnInitDialog()*, copiando la ruta del fichero en el parámetro *nomfichcats* de la variable **ComprensionLineal**. La función encargada de cargar en **ComprensionLineal** todas las glosas con su categoría correspondiente es *lee_categorias*. Esta función se encarga de rellenar **ComprensionLineal**:

- **ComprensionLineal->palabras:** va cargando cada una de las palabras de *categorías_lineal.txt*.

- **ComprensionLineal->categorias:** en principio este parámetro está pensado para poder cargar varias categorías por cada **palabra**, aunque en este caso sólo cargamos una categoría por cada **palabra**. El primer índice de **categorias** se corresponde con el índice de **palabra**.

En este caso únicamente indicamos una categoría por cada palabra. Para separar la palabra de su categoría utilizamos el carácter ‘1’ en el fichero *categorias_lineal.txt*. En la primera línea del fichero indicamos el número de palabras existentes en el mismo. El aspecto del fichero es el siguiente:

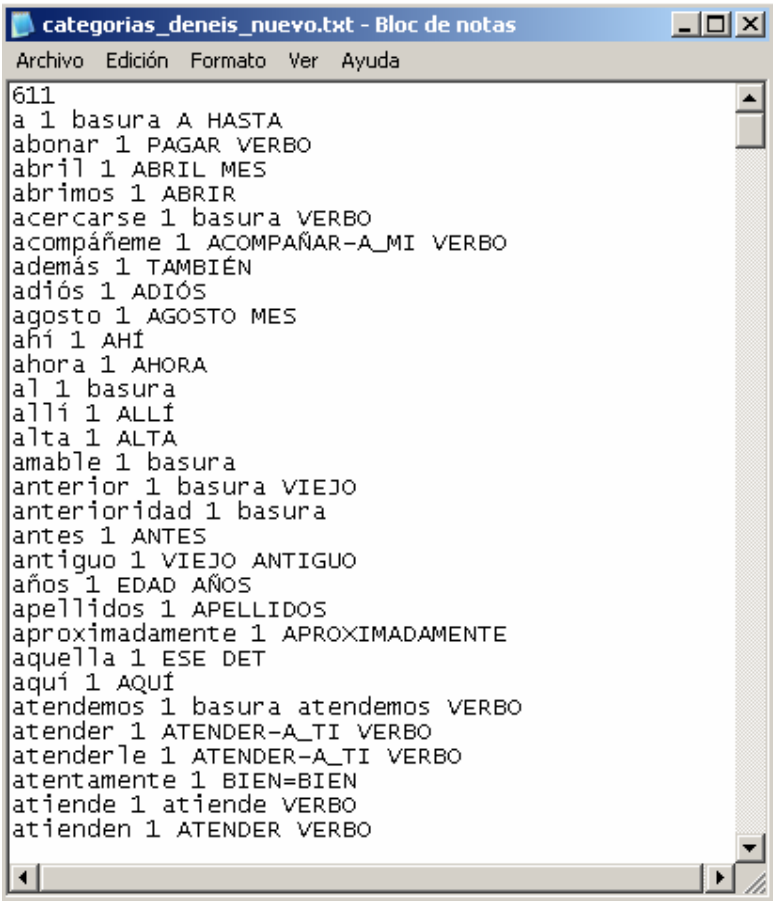


Figura 21: Fichero *categorias_lineal.txt* donde indicamos la categoría de cada palabra

Una vez inicializada la variable **ComprensionLineal**, sólo queda comentar el tratamiento de la traducción en sí, que llamamos cada vez que actualizamos la cadena de signos a traducir, mediante la función *traduceLineal()*.

Las operaciones básicas realizadas en esta función son:

- Cargamos cada palabra de la frase a traducir para procesarlos y traducirlos.
- Procesamos primero la cadena a traducir, categorizando cada palabra.
- Calculamos la mínima distancia entre la frase a traducir y los ejemplos cargados de la base de datos.
- Devolvemos la distancia calculada.

Tabla 4: Esquema de *traduceLineal()*

Las palabras de la frase las vamos cargando en la variable **ComprensionLineal**, más concretamente en **ComprensionLineal->bloques**. Las palabras cargadas en este parámetro serán las que procesemos posteriormente para traducirlas.

La función encargada de categorizar estas palabras, cargadas en **bloques**, es *preprocesaLineal*. Esta función se encuentra en el fichero *comprensión.cpp*, categorizando las palabras a traducir de la siguiente manera:

- Para cada palabra cargada en **bloques**:
  - La comparamos con las palabras cargadas en **palabras**:
    - Si la palabra se encuentra en **palabras**:
      - Copiamos su categoría (cargada en **categorias**) en **etiquetas**
    - Si la palabra no se encuentra en **palabras**:
      - Copiamos en **etiquetas** la categoría “basura”

Tabla 5: Esquema de *preprocesaLineal()*

El primer índice de **etiquetas** se corresponde con la palabra a traducir (índice de **bloques**). Al igual que pasaba con el parámetro **categorias**, en **etiquetas** podemos insertar varias categorías (o etiquetas), pero en este caso únicamente indicamos una categoría.

Si la palabra a traducir no se encuentra en **palabras** (lo que quiere decir que esa palabra no aparece en los ejemplos de la base de datos) le asignamos la categoría $DELETREO, ya que en principio suponemos que es un nombre propio (o quizá un signo correspondiente a otro ámbito de trabajo, no al utilizado en el proyecto). Dentro de los ejemplos, existen frases preparadas para indicar nombres propios, por lo que es necesario indicar esta categoría, ya que en principio debería tratarse de una frase de este tipo.

Para calcular la distancia entre la frase a traducir y los ejemplos de la base de datos, hacemos uso de la función *procesaLineal*, a la que debemos pasarle la variable **lista**, donde hemos cargado al inicio de la aplicación cada uno de los ejemplos existentes en la base de datos.

El cálculo de la distancia entre la frase a traducir y los ejemplos de la base de datos necesita de una previa conversión de la frase a traducir en sus valores numéricos. La conversión se realiza identificando las palabras a traducir entre las palabras de **lista_pal**. Para calcular el índice correspondiente a cada glosa dentro de **lista_pal** utilizamos la función *IndicePalabra*, guardando todos ellos en **frase_cat**. Para calcular la distancia utilizamos la frase a traducir categorizada, por lo que los índices que debemos calcular corresponden a las glosas guardadas en **etiquetas** (el motivo de utilizar índices en lugar de las palabras es para que el algoritmo sea más rápido y nos permita comparar un mayor número de ejemplos). Una vez convertida la frase en los índices correspondientes, calculamos la distancia:

- Para cada ejemplo de la lista:
  - Calculamos la distancia entre el ejemplo y la frase a traducir
  - Si la distancia es menor que **distancia_min**, actualizamos ésta con el nuevo valor, convirtiendo el ejemplo actual en el mejor (*struct **param*** **lista_mejor**)
  - Pasamos al siguiente ejemplo (**lista->next**)

- Calculamos la distancia relativa, dividiendo entre el número de glosas a traducir
- Para cada palabra de **lista_mejor:**
    - Si corresponde a una categoría:
        - Añadimos a la salida la palabra original (**bloques**) en mayúsculas (paso a glosas)
    - Si no corresponde a una categoría:
        - Añadimos a la salida la glosa de **lista_mejor**
- Para cada glosa guardada en la salida (hemos ido guardando en la variable **salida**):
    - Copiamos la glosa de **salida** en **etiquetas**
- Para cada glosa copiada en **etiquetas**:
    - Copiamos la palabra en **ComprensionLineal->vector_obs_ATRIBUTO**

Tabla 6: Esquema de *procesaLineal()*

Para calcular la distancia entre la frase a traducir categorizada y cada ejemplo de la base de datos, utilizamos la función *Distancia*. A esta función le pasamos los índices correspondientes a la frase a traducir categorizada, además de los índices correspondientes al ejemplo, que también está categorizado, por lo que debemos pasarle la variable **ComprensionLineal->p_e_cat**.

El ejemplo categorizado que más se parece a la frase a traducir categorizada lo guardamos en **lista_mejor**. Al terminar el cálculo de la distancia con todos los ejemplos, debemos guardar el resultado del mejor ejemplo (**lista_mejor**) en **ComprensionLineal**. La traducción categorizada del mejor ejemplo se encuentra en **lista_mejor->p_s_cat**, de donde iremos obteniendo la palabra correspondiente de **lista_pal** (en **p_s_cat** guardamos los índices de las palabras). En alguna de estas palabras, resultado de la traducción, podemos encontrarnos alguna de ellas categorizada, por lo que habrá que sustituirla por su correspondiente traducción. En nuestro caso, esta traducción corresponde únicamente a la palabra original (que habremos categorizado al comienzo del proceso) en mayúsculas (paso a glosas). Si por ejemplo la glosa que aparece en **p_s_cat** es **$ORDINAL**, y en la frase original a traducir se encontraba en su misma posición la palabra **segundo**, en la salida debemos copiar "**SEGUNDO**".

Este proceso de sustitución de las categorías que aparecen en el ejemplo escogido como traducción sigue las siguientes normas:

- Si en el ejemplo hay dos categorías iguales (como por ejemplo $NUMERO), la traducción de texto a LSE (paso a mayúsculas) se hace por orden de aparición de las categorías en la frase original.

- Si el ejemplo tiene una categoría que no aparece en la frase original, dicha categoría no se puede rellenar y por tanto no se incorpora en la traducción final.

El resultado de la traducción del mejor ejemplo (**lista_mejor->p_s_cat**) lo vamos copiando en el array **salida**, sustituyendo adecuadamente, como hemos comentado, las glosas que fueran una categoría. Una vez copiado el resultado y tratado las posibles categorías, debemos actualizar la variable **ComprensionLineal**, copiando para ello el resultado en **ComprensionLineal->etiquetas**. Posteriormente, copiamos a su vez este resultado en **ComprensionLineal->vector_obs_ATRIBUTO**, utilizando para ello la función *comprensión*. Esta última copia también se realiza en la traducción basada en reglas, dejando también el resultado en **vector_obs_ATRIBUTO**. Debido a este hecho, para unificar la obtención de la salida en *traduce()*, realizamos esta copia doble.

La función procesaLineal devuelve la distancia relativa obtenida del mejor ejemplo. Para calcular la distancia relativa dividimos la distancia obtenida en Distancia entre el número de palabras de la frase a traducir. De esta manera obtenemos una distancia independiente del número de palabras que estemos traduciendo, pudiendo así establecer un umbral de forma más adecuada. Este umbral se denomina UMBRAL_DISTANCIA, y lo utilizamos en la función traduce() para utilizar el resultado de una u otra traducción.

Para establecer este umbral hemos partido del valor medio de las distancias mínimas entre ejemplos dentro de la base de datos disponible. Para calcular este valor hemos ido calculando la distancia relativa mínima de cada ejemplo de la base de datos con el ejemplo más parecido. El valor medio obtenido de todas estas distancias relativas mínimas ha sido de 0.513508. Una vez obtenido este valor, hemos establecido el valor justo por debajo como umbral, esto es 0.51. Este valor lo hemos ido ajustando empíricamente, probando cuándo se realizaba cada una de las traducciones en el sistema y comprobando los resultados obtenidos. El valor final establecido para el umbral ha sido el de 0.4, es decir un 40%.

6.2. Traducción basada en reglas

El método de traducción implementado en el sistema de traducción de signo-escritura a voz en castellano, es un método de traducción basado en reglas que mezcla métodos directos e indirectos. Algunas de las reglas aplicadas en el sistema transforman una palabra en castellano en una glosa en LSE, equivalente a un método directo de traducción bilingüe palabra por palabra. La mayor parte de las reglas aplicadas son de tipo sintáctico-semánticas, que van transformando la secuencia origen hasta convertirla en la secuencia resultado de la traducción, por lo que el método utilizado principalmente es el de traducción por transferencia. Las reglas aplicadas están fuertemente ligadas con las lenguas origen y destino, castellano y Lengua de Signos Española representada en glosas. Estas reglas, además, son dependientes del entorno de trabajo del sistema. Los dos entornos

desarrollados en este proyecto son el trámite administrativo de recogida y renovación del DNI y del carné de conducir.

El método implementado se divide principalmente en dos fases, que describimos a continuación:

- La primera fase corresponde a la de categorización de la secuencia de entrada. A cada una de las palabras de la secuencia en la lengua origen (castellano) se le asignan una o varias categorías sintáctico-semánticas, que serán necesarias para poder aplicar correctamente las reglas.

- En la segunda fase es donde se aplican estas reglas, que diferenciarán las palabras dependiendo de las categorías que tengan asignadas. Estas reglas se van aplicando sucesivamente sobre bloques de la secuencia de entrada (los bloques originarios son las palabras originales junto con las categorías que tengan asignadas, y van cambiando según se van aplicando cada una de las reglas), hasta llegar a los bloques finales, que conformarán la secuencia de salida, en la lengua destino. Las reglas se implementan en un lenguaje propietario formado por un conjunto de primitivas.

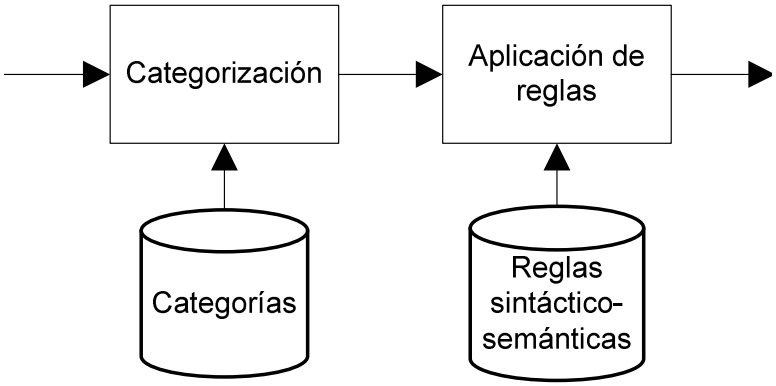


Figura 22: Método de transferencia implementado en el sistema

6.2.1. Categorización de las palabras de entrada

En relación con la categorización del texto cabe comentar que se realiza en base a un fichero de texto que va asociando a cada palabra origen, un conjunto de categorías que podrán ser conceptos intermedios, o en algunos casos, palabras de la lengua destino (LSE).

En relación con este proceso de categorización, cabe resaltar las siguientes características:

- En primer lugar, comentar que algunas de las palabras se han categorizado como “basura”. Estas palabras, ni generan signos en destino ni son utilizados por ninguna de las reglas intermedias. Las palabras categorizados con “basura” se eliminarán de la secuencia de bloques durante el proceso de traducción.

- En segundo lugar hay que decir que el criterio que hemos seguido para la generación de las palabras en el idioma destino es que la PRIMERA CATEGORÍA de cada bloque contendrá, al final del proceso de traducción, las palabras de la lengua destino (en este caso signos de LSE). El resto de categorías aportarán información relevante para el proceso de traducción (aplicación de las reglas). A continuación veremos un ejemplo de la categorización de la palabra “dónde”:

  dónde 1 DÓNDE? PART_INTERROG

  La palabra “dónde” se categoriza con el signo “DÓNDE?” que será la palabra en el idioma destino y con la categoría “PART_INTERROG” que nos dice que es una partícula interrogativa, y que por tanto, debe ir al comienzo de la frase (información importante para las reglas de ordenación).

- En el caso de que una palabra genere varios signos (glosas), estos se considerarán como si fueran un único elemento, uniendo los segmentos con caracteres de igual.

  rellene 1 TU=ESCRIBIR

  Al final del proceso de traducción se dividirán los signos en bloques independientes (utilizando la primitiva partir_todas_separadores).

- Por último, hay que comentar para el caso de aquellas palabras que no aparezcan en el fichero de categorías, NO se etiquetarán con la categoría “basura” sino con la propia palabra (pasada a mayúsculas). De esta forma queremos resolver el problema de los nombres propios. Los nombres propios en castellano y LSE se escriben igual luego la traducción debe ser literal. Finalmente, el sistema de representación, si no dispone de un signo específico para representar ese nombre lo deletreará.

6.2.2. Primitivas

Las reglas que se han diseñado en un lenguaje formado por un conjunto de primitivas. Las primitivas utilizadas en nuestro caso son 10 y se describen a continuación:

reescribe(N,M,"catN1","catN2",..,”catM1”,”catM2”,..,Comprension);

Esta primitiva permite especificar una secuencia consecutiva de bloques que tienen las N categorías “catN1”,.., “catNN” que se sustituirá por una nueva secuencia de M bloques que tendrán las categorías “catM1”,..,”catMM”.

recat(N,M,"catN1","catN2",..,”catM1”,”catM2”,..,Comprension);

Esta primitiva permite especificar un único bloque que tiene las N categorías “catN1”,.., “catNN” que se sustituirá por otro único bloque con M categorías “catM1”,..,”catMM”.

pon_delante_o_primero("cat1","cat2",Comprension);

Esta primitiva busca en la secuencia de bloques un bloque categorizado con “cat1” y va recorriendo los bloques anteriores hasta encontrar un bloque categorizado con “cat2”. Si lo encuentra, coloca el bloque con “cat1” delante del bloque con “cat2”. Si no encuentra ningún bloque con “cat2” colocará el bloque con “cat1” al comienzo de todo.

pon_detras_o_ultimo("cat1","cat2",Comprension);

Esta primitiva busca en la secuencia de bloques un bloque categorizado con “cat1” y va recorriendo los bloques posteriores hasta encontrar un bloque categorizado con “cat2”. Si lo encuentra, coloca el bloque con “cat1” detrás del bloque con “cat2”. Si no encuentra ningún bloque con “cat2” colocará el bloque con “cat1” al final de todos.

elimina_basura(Comprension);

Elimina de la secuencia de bloques todos aquellos que tengan la categoría “basura”.

anyade_cats_si(N,M,"catN1","catN2",.,”catM1”,”catM2”,.,Comprension);

Esta primitiva permite especificar una secuencia consecutiva de bloques que tienen las N categorías “catN1”,.., “catNN”. Si se encuentra esta secuencia de categorías, a todas ellas se les añadirán M categorías: “catM1”,..,”catMM”.

int Existe(“cat1”,Comprension);

Esta primitiva devuelve un 1 si la categoría “cat1” está en alguno de los bloques y 0 si no está en ningún bloque.

Elimina(“cat1”,Comprension);

Esta primitiva elimina todos los bloques que estén categorizados con la categoría “cat1”. Esto sólo se utiliza para eliminar las formas verbales del verbo SER que en Lengua de Signos Española se omiten.

quita_repes(“cat1”,Comprension);

Esta primitiva busca dos bloques consecutivos categorizados con la categoría “cat1” y elimina el segundo de ellos.

partir_todas_separadores("XXXXX",Comprension);

Todos aquellos bloques en los que la primera de las categorías (que como habíamos comentado, contiene las palabras en la lengua destino unidas por guiones) tenga varias palabras unidades por alguno de los caracteres indicados en la secuencia XXXXXX, cada una de las palabras se guardará en un bloque diferentes. De forma que, por ejemplo en el caso de considerar el guión como un posible separador, un bloque categorizado con “el-deneí” se convertirá en dos bloques con las categorías “el” y “deneí”.

6.2.3. Reglas de traducción

En este apartado comentaremos a nivel general los tipos de reglas utilizados y los principales problemas de traducción que se pretenden resolver.

Las reglas implementadas se pueden estructurar en 5 grandes grupos.

El primer bloque de reglas pretende realizar las siguientes tareas:

- Tareas de traducción de diferentes perífrasis verbales, como por ejemplo el siguiente caso:

reescribe(2,1,"deben","ser","DEBER",Comprension);

La secuencia de palabras “deben ser” se traduce en LSE en el signo “DEBER”.

- Uso de palabras cortas (“sus” en el ejemplo) como apoyo (sabiendo que esto puede ser peligroso en el caso de fallos del reconocedor) ya que en ocasiones ayudan a separar bloques de información. Un ejemplo es:

reescribe(2,1,"sus","DATO","TU=DATOS-PERSONALES",Comprension);

La secuencia “sus DATO” se traducirá en un único bloque TU=DATOS-PERSONALES que posteriormente dará lugar a dos bloques TU DATOS-PERSONALES mediante la función **partir_todas_separadores.**

- Borrado de algunas secuencias de bloques.

reescribe(3,1,"TENER","QUE","SACAR","basura",Comprension);

La secuencia de bloques con las categorías TENER QUE SACAR se convierte en basura que luego se eliminará

- Se procesan otros elementos como por ejemplo:

reescribe(2,1,"MUCHAS","GRACIAS","MUCHAS-GRACIAS",Comprension);

- Se reutilizan los elementos procedentes de las perífrasis verbales junto con otros signos para producir signos de mayor complejidad, como por ejemplo:

reescribe(2,1,"TU=ESCRIBIR=DEBER","PAPEL=ESTE","PAPEL=ESTE=TU=ESCRIBIR=DEBER",Comprension);

Seguidamente se llama a la regla **elimina_basura** que elimina los bloques categorizados como basura.

El segundo grupo de reglas realizan las labores:

- Combinación de bloques después de haber eliminado las basuras. Como por ejemplo:

reescribe(3,2,"TENER","BUENOS","DÍAS","PASAR","BIEN",Comprension);

Reescribe los bloques TENER BUENOS DÍAS por los bloques PASAR BIEN

- Preparación y agrupamiento de bloques para su posterior ordenamiento. Por ejemplo:

anyade_cats_si(3,1,"MÁS","NÚMERO","EDAD","COMPLEMENTO",Comprension);

A la secuencia de bloques MÁS NÚMERO EDAD les añade la categoría COMPLEMENTO para después ordenar el COMPLEMENTO.

- Se utilizan elementos procedentes de otras reglas para generar signos más complejos:

  reescribe(3,1,"HACER-FALTA-NO","RENOVAR","DNI","DNI=RENOVAR=HACER-FALTA-NO",Comprension);

- Se termina de procesar expresiones que no se hubiesen procesado antes del proceso de eliminación de las palabras “basura”.

En tercer lugar se encuentran las reglas de ordenamiento donde las primitivas **pon_delante_o_primero** y **pon_detras_o_ultimo** se usan masivamente:

- Se ordenan signos que no tienen un orden sujeto al contexto, por ejemplo el signo “TU” siempre ocupa el primer lugar en la frase:

  pon_delante_o_primero ("TU","-",Comprension);

- Se ordenan elementos que están sujetos a un contexto, es decir su posición está ligada a la existencia de determinados signos:

  pon_delante_o_primero ("MANO=DERECHA","DEDO",Comprension);

- Se colocan los elementos que van en posiciones finales o detrás de otros elementos:

  pon_detras_o_ultimo ("DEBER","-",Comprension);

El cuarto bloque lo forman un conjunto de reglas que resuelven casos muy concretos y que no son para nada generales pero asociadas directamente a la tarea del trámite del DNI o del carné de conducir. Un ejemplo es:

```
If ((Existe("CATORCE",Comprension) && Existe("OBLIGATORIO",Comprension)))
    reescribe(1,2,"OBLIGATORIO","DNI","OBLIGATORIO",Comprension);
```

En el caso de que exista el bloque CATORCE y OBLIGATORIO hay que introducir el DNI antes de OBLIGATORIO porque en este contexto se está haciendo referencia al DNI.

Finalmente, en **el bloque quinto** se sitúan reglas de ajustes finales, y son las siguientes:

- Elimina los bloques con categoría SER

  Elimina("SER",Comprension);

- Se quitan repetidos.

  quita_repes ("TU",Comprension);

  quita_repes ("DNI",Comprension);

  quita_repes ("MÁS",Comprension);

- Se separan los bloques con “=” en varios bloques:

partir_todas_separadores("=",Comprension);

Con esta regla separamos los bloques que tengan varios signos en LSE separadas por un igual. De esta forma se genera la secuencia de bloques final que contiene, en la primera de las categorías de cada uno de los bloques, un único signo.

6.2.4. Estadísticas

Para terminar a modo de resumen se comentan los principales números del proceso de traducción en el caso de renovación del DNI:

- Número de frases a traducir: 475. Consideradas en el estudio para el desarrollo de las reglas.
- Número de palabras en origen (palabras en Castellano): 2743
- Número de palabras en destino (signos): 2085
- Número de primitivas utilizadas: 10
- Número de reglas total: 293
    - reescribe: 180
    - recat: 3
    - pon_delante_o_primero: 13
    - pon_detras_o_ultimo: 74
    - elimina_basura: 1
    - anyade_cats_si: 2
    - Existe: 15
    - Elimina: 1
    - quita_repes: 3
    - partir_todas_separadores: 1

Estos números son muy similares a los obtenidos para el dominio de renovación del carné de conducir.

6.2.5. Resultados

Aunque la tasa de acierto de palabra ha sido muy baja (61%), se han comprobado manualmente las frases (475) y con la versión actual en más del 80% de los casos se ha generado una frase coherente y con el significado adecuado. Todavía falta seguir trabajando en las reglas porque esta diferencia entre el porcentaje de palabras correctas y las frases generadas nos da una idea de la gran variabilidad que tenemos en la traducción de las frases: estructura, etiquetado múltiple de las glosas (varias glosas para el mismo signo), sinónimos de signos o utilización de perífrasis (DATOS por NOMBRE DIRECCIÓN).

6.3. Traducción estadística

La traducción estadística consiste en un algoritmo de búsqueda dinámica que utiliza un modelo estadístico para obtener la mejor secuencia de signos resultado de la traducción de una secuencia de palabras obtenidas del reconocedor de voz. Este modelo integra principalmente información de dos tipos de probabilidades:

- Probabilidad de traducción: recoge información sobre qué palabras se traducen por qué signos.
- Probabilidad de la secuencia de signos: aporta información sobre qué secuencias de signos son más probables en la LSE.

En este paso se realiza una traducción de las palabras provenientes del reconocedor a signos correspondientes, en este caso, a la Lengua de Signos Española. Para esto se utilizan métodos estadísticos cuyos Modelos se aprenden a partir de un corpus paralelo, compuesto por documentos de texto en castellano y sus equivalentes en Lengua de Signos. El documento de texto contendrá palabras en castellano, mientras que el de LSE contendrá GLOSAS. Las glosas son palabras (en mayúsculas) que representan los signos. Por ejemplo la glosa FOTO representa el signo cuyo significado es el de "fotografía".

6.3.1. Traducción Estadística basada en Modelos de Subsecuencias de Palabras

La traducción estadística basada en modelos de subsecuencias de palabras (o subfrases) consiste en la obtención de un Modelo de Traducción a partir del alineamiento y extracción de subsecuencias utilizando un corpus paralelo, y la generación de un modelo de lenguaje de la lengua destino. Estos modelos se utilizan por el módulo de traducción (Moses) para obtener la secuencia de signos/glosas dada una frase de entrada. La arquitectura completa de este sistema de traducción es la siguiente:

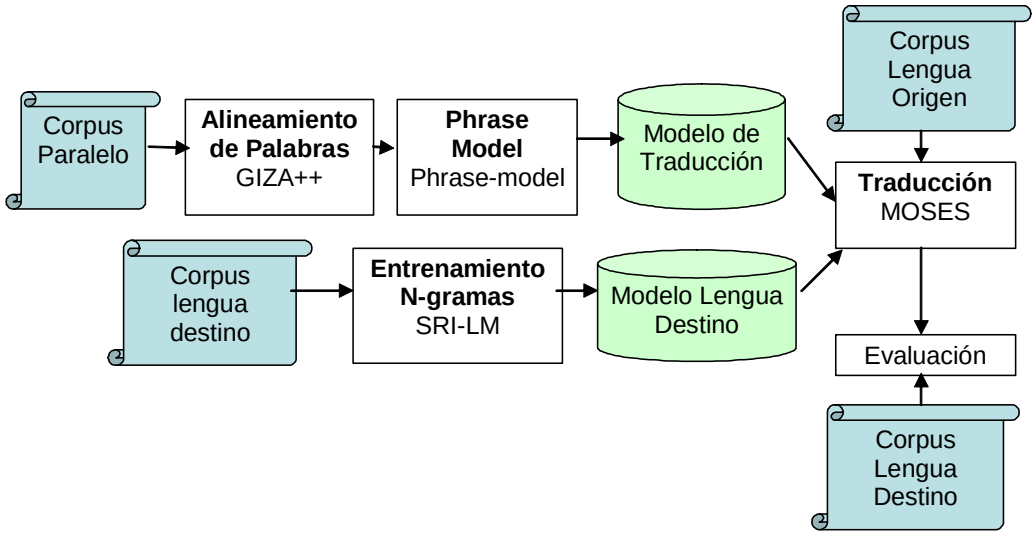


Figura 23. Arquitectura de la Traducción basada en Subsecuencias de Palabras

6.3.1.1 Generación de Modelos

En primer lugar debe crearse el Modelo de Lenguaje (de la lengua destino) y el Modelo de Traducción (a partir de un corpus paralelo tanto en lengua origen como destino). Las ideas que hay detrás de la traducción automática estadística vienen de la teoría de la información. Esencialmente, el problema de la traducción se centra en conocer la probabilidad p(d|o) de que una cadena o de la lengua origen genere una cadena d en la lengua destino. Estas probabilidades se calculan utilizando técnicas de estimación de parámetros a partir del corpus paralelo.

Aplicando el Teorema de Bayes a p(d|o) se puede representar esta probabilidad como el producto p(o|d)·p(d), donde el Modelo de Traducción p(o|d) es la probabilidad de que la cadena origen se traduzca por la cadena destino, y el Modelo de Lenguaje p(d) es la probabilidad de ver aquella cadena origen. Matemáticamente, encontrar la mejor traducción $\tilde{o}$ se consigue escogiendo aquella secuencia de signos que permita obtener la probabilidad máxima:

$$\tilde{o} = \arg\max_{o \in o^*} p(d/o) = \arg\max_{o \in o^*} p(o/d) \cdot p(d) \qquad (1)$$

Para la creación del Modelo de Lenguaje, se utiliza la herramienta SRILM (Stolcke, 2002), una herramienta que realiza la estimación de los modelos de lenguaje tipo N-grama, a partir del corpus de entrenamiento, y su evaluación calculando la probabilidad de un corpus de test. Estos Modelos se utilizan ampliamente en muchos ámbitos: reconocimiento de habla, OCR (Reconocimiento Óptico de Caracteres), etc. En cuanto a los Modelos de Traducción, su generación se hace mediante una traducción basada en subfrases. Para esto la herramienta utilizada es el GIZA++ (que es una implementación de los modelos IBM de traducción), un sistema de traducción estadística automática capaz de entrenar estos modelos para cualquier par de lenguas (http://www.statmt.org/moses). Para esto se necesita una colección de textos traducidos, que será el corpus paralelo. Los pasos para la generación de los modelos son:

1. Obtención del alineamiento entre palabras: consiste en que a partir de los dos textos en castellano y LSE se identifican qué palabras de uno corresponden con las del otro. Para esto se utiliza el programa GIZA++. El alineamiento se hace tanto en el sentido palabras-glosas como en la dirección glosas-palabras. Un ejemplo de un alineamiento es el siguiente:

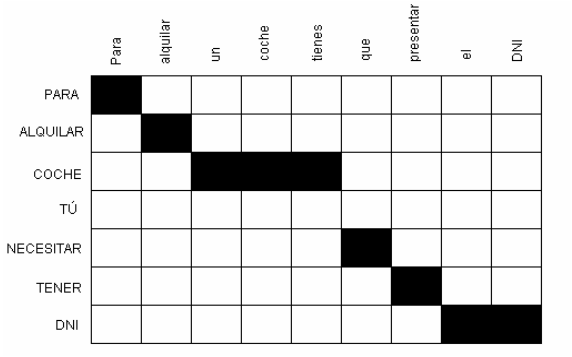


Figura 24. Ejemplo de un alineamiento entre palabras en castellano y signos en LSE representados mediante glosas.

2. Cálculo de una tabla de traducción léxica: a partir del alineamiento, se realiza una estimación de la tabla de traducción léxica más probable, obteniendo los valores de $w(d|o)$ y su inversa $w(o|d)$ para todas las palabras, es decir, las probabilidades de traducción para todos los pares de palabras. Un ejemplo para la palabra "por" con el texto utilizado es:

por PRIMER 0.5000000

por POR 0.3333333 …

3. Extracción de subsecuencias de palabras: se recopilan todos los pares de subsecuencias que sean consistentes con el alineamiento. El archivo generado en este paso tiene la forma siguiente, donde la subfrase "a los siguientes países" se traduce por la subsecuencia de glosas "ESTOS PLURAL PAÍS":

a los siguientes países ||| *ESTOS PLURAL PAÍS* ||| *0-0 2-0 1-1 3-2*

a los siguientes ||| *ESTOS PLURAL* ||| *0-0 2-0 1-1*

4. Cálculo de las probabilidades de traducción de cada subsecuencia ("Phrase Scoring"): En este paso, se calculan las probabilidades de traducción para todos los pares de subfrases en los dos sentidos: subfrase en castellano-signo en LSE y signo en LSE – subfrase en castellano. Un ejemplo del archivo obtenido es:

a los siguientes países ||| *ESTOS PLURAL PAÍS* ||| *(0) (1) (0) (2)* ||| *(0,2) (1) (3)* ||| *1 0.0283293*

a los siguientes ||| *ESTOS PLURAL* ||| *(0) (1) (0)* ||| *(0,2) (1)* ||| *1 0.0661018*

6.3.1.2 Ajuste

Para realizar el proceso de traducción se deben combinar los modelos generados en la fase anterior de entrenamiento. Esta composición se hace mediante una combinación lineal de probabilidades cuyos pesos se deben ajustar. El proceso de ajuste de los pesos consiste en probar el traductor Moses con un conjunto de frases (conjunto de validación) y, conociendo la traducción correcta, evaluar las salidas del traductor automático en función de los valores diferentes asignados a los pesos. Estos valores se eligen

aleatoriamente y después de una búsqueda también aleatoria se eligen los valores que hayan ofrecido los mejores resultados.

6.3.1.3 Traducción

Utilizando un nuevo conjunto de frases (conjunto de test) se evalúa el sistema. Tanto para la fase de ajuste como para la de evaluación se utiliza el traductor Moses que emplea los modelos obtenidos anteriormente (modelo de traducción y modelo de lenguaje de la lengua destino), combinados según los pesos ajustados. Moses (http://www.statmt.org/moses) es un sistema de traducción automática estadística basado en subsecuencias de palabras, que implementa un algoritmo de búsqueda para obtener, a partir de una frase de entrada, la secuencia de signos que con mayor probabilidad corresponde a su traducción. Permite trabajar con redes de confusión de palabras como las que se obtienen en gran cantidad de sistemas de reconocimiento de voz. Por otro lado, también permite la integración de varios modelos de traducción entrenados con los diferentes factores con los que se puede etiquetar las palabras de las frases.

6.3.2. Traducción Estadística basada en Transductores de Estados Finitos

Los transductores de Estados Finitos ("FSTs: Finite State Transducers") se están usando en diferentes áreas de reconocimientos de patrones y lingüística computacional. Los FSTs parten de un corpus de entrenamiento que consta de pares de frases origen-destino, y usando métodos de alineamiento basados en GIZA++ generan un conjunto de cadenas a partir de las cuales se puede inferir una gramática racional. Esta gramática se convierte, por último, en un traductor de Estados Finitos. Una de las principales razones del interés de esta técnica es que las máquinas de estados finitos pueden aprenderse automáticamente a partir de ejemplos.

Un FST se caracteriza por la topología y por las distribuciones de probabilidad, dos características distintivas que se pueden aprender de un corpus bilingüe mediante algoritmos eficientes, como el GIATI ("Grammar Inference and Alignments for Transducers Inference"). En la figura siguiente se muestra la arquitectura de esta solución. Los pasos de esta estrategia de traducción son los que se explican a continuación.

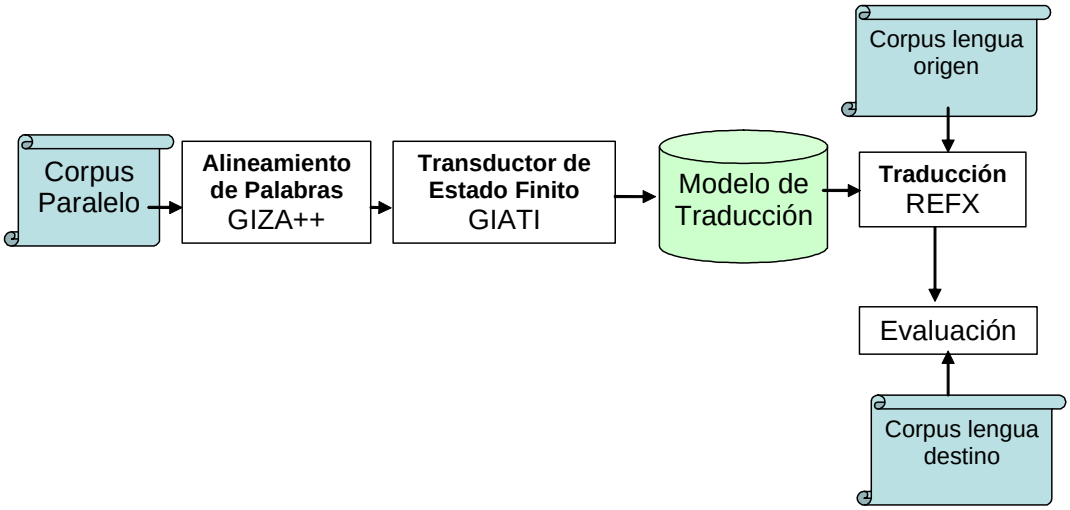


Figura 25: Arquitectura de la Traducción basada en Transductores de Estados Finitos

6.3.2.1 Alineamiento con GIZA++

En esta fase se pretende el alineamiento de las palabras de las frases de entrada (en castellano) con los signos/glosas de sus traducciones correspondientes (en LSE). Este alineamiento se realiza en los dos sentidos: tanto en el sentido palabras-glosas como en la dirección glosas-palabras. Para realizar este alineamiento se utiliza el programa GIZA++ como se comentó anteriormente.

6.3.2.2 Transformación de pares de entrenamiento a frases de palabras extendidas

Partiendo de un alineamiento como el explicado en la sección 3.1, se realiza un proceso de etiquetado, en el cual se construyen un corpus extendido a partir de cada uno de los pares de subsecuencias de entrenamiento y sus correspondientes alineamientos: se asignarán por tanto palabras de lengua origen a su correspondiente palabra en lengua destino gracias a su alineamiento. Se muestra a continuación un ejemplo de pares castellano / LSE (en glosas) y su alineamiento:

el denei es obligatorio desde los catorce años # DNI(2) SE-LLAMA(3) OBLIGATORIO(4) DESDE(5) CATORCE(7) PLURAL(6) AÑO(8) EDAD(8)

el denei es obligatorio # DNI(2) SE-LLAMA(3) OBLIGATORIO(4)

el denei es el documento oficial # DNI(2) SE-LLAMA(3) DOCUMENTO(5) OFICIAL(6)

el denei es oficial # DNI(2) SE-LLAMA(3) OFICIAL(4)

Para que no se produzca una violación en el orden secuencial de las palabras en la lengua destino, se sigue el siguiente criterio de etiquetado: cada palabra de lengua destino se une con su correspondiente palabra en lengua origen a partir del alineamiento si el orden de las palabras objetivo no se altera. Si fuera así, la palabra en lengua destino se une con la primera palabra en lengua origen que no viole el orden de las palabras objetivo. Por lo tanto, el ejemplo anterior quedaría de la siguiente manera, con la formación de palabras extendidas ("extended words", unión de palabras y signos alineados):

(el, λ) (denei, DNI) (es, SE-LLAMA) (obligatorio, OBLIGATORIO) (desde, DESDE) (los, PLURAL), (catorce, CATORCE) (años, AÑO EDAD)

(el, λ) (denei, DNI) (es, SE-LLAMA) (obligatorio, OBLIGATORIO)

(el, λ) (denei, DNI) (es, SE-LLAMA) (el, λ) (documento, DOCUMENTO) (oficial, OFICIAL)

(el, λ) (denei, DNI) (es, SE-LLAMA) (oficial, OFICIAL)

Si se hace un refinamiento de este etiquetado, se puede implementar que las palabras origen que hayan quedado aisladas se unan a la primera palabra extendida que tenga palabra(s) destino asignadas. Por lo tanto, el ejemplo anterior se convierte en:

(el denei, DNI) (es, SE-LLAMA) (obligatorio, OBLIGATORIO) (desde, DESDE) (los, PLURAL), (catorce, CATORCE) (años, AÑO EDAD)

(el denei, DNI) (es, SE-LLAMA) (obligatorio, OBLIGATORIO)

(el denei, DNI) (es, SE-LLAMA) (el documento, DOCUMENTO) (oficial, OFICIAL)

(el denei, DNI) (es, SE-LLAMA) (oficial, OFICIAL)

6.3.2.3 Inferencia de un Gramática Estocástica y posteriormente de un Traductor de Estados Finitos

Consiste en la obtención de un Transductor de Estados Finitos a partir de las frases con las palabras extendidas. Las probabilidades de saltos entre nodos de un FST se computan por las cuentas correspondientes en el conjunto de entrenamiento de palabras extendidas. La probabilidad de una palabra extendida zj a partir de una palabra origen si y una palabra destino ti: zj = (si, ti), dada una secuencia de palabras extendidas zi−n+1, zi−1 = (si−n+1,ti−n+1) (si−1,ti−1) es:

$$p_n(z_i \mid z_{i-n+1} \ldots z_{i-1}) = \frac{c(z_{i-n+1}, \ldots, z_{i-1}, z_i)}{c(z_{i-n+1}, \ldots, z_{i-1})} \qquad (2)$$

Donde c(·) es el número de veces que ocurre un evento en el conjunto de entrenamiento. A partir del resultado del apartado anterior se infiere un modelo tipo bigrama. Se ilustra este proceso en la siguiente figura, donde los nodos grises indican que la subfrase puede terminar en ese punto:

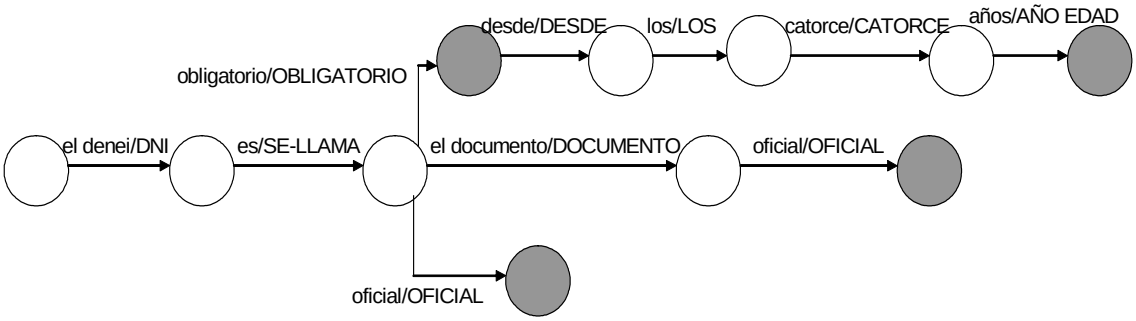


Figura 26: Transductor de estado finito inferido a partir del bigrama del ejemplo anterior

6.3.3. Evaluación de la traducción estadística

6.3.3.1 Medidas de evaluación

Con el objetivo de evaluar la calidad de la traducción, es necesario comparar la salida del sistema automático con una referencia y calcular algunas medidas de evaluación. WER ("Word Error Rate", porción de palabras con error) es una medida comúnmente utilizada en la evaluación de sistemas de reconocimiento del habla o de traducción automática. Calcula el número de inserciones, borrados y sustituciones de palabras cuando se comparan frases. Esta medida se basa en la distancia de edición o de Levensthein. En tareas tanto de traducción automática como de reconocimiento del habla, se calcula el WER entre la frase generada por el sistema de traducción y una frase que es de referencia correcta (en este caso, frases de signos o glosas).

BLEU (Bilingual Evaluation Understudy) (Papineni et al, 2002) es un método de evaluación de la calidad de traducciones realizadas por sistemas de traducción automática. Una traducción tiene mayor calidad cuanto más similar es con respecto a otra de referencia, que se supone correcta. BLEU puede calcularse utilizando más de una traducción de referencia. Esto permite una mayor robustez a la medida frente a traducciones libres realizadas por humanos. BLEU se calcula normalmente a nivel de frases y halla la precisión en n-gramas entre la traducción del sistema y la de referencia. Estas medidas surgen con el objetivo de encontrar medidas automáticas que correlen con la evaluación que un experto haría de la traducción.

Otra medida es NIST, que se basa en la BLEU con algunas modificaciones: en primer lugar, BLEU utiliza la media geométrica de la precisión de los N-gramas, mientras que NIST utiliza una media aritmética para reducir el impacto de bajas concurrencias para órdenes altos de n-gramas. También, BLEU calcula la precisión de n-gramas utilizando pesos iguales para cada n-grama, mientras que NIST considera la calidad de la información que proporciona un n-grama particular en sí mismo (por ejemplo, cuanto menos frecuente sea un n-grama más peso se le asignará).

6.3.3.2 Base de Datos

La base de datos utilizada para los experimentos consiste en un corpus paralelo que contiene 414 frases típicas de un contexto restringido: aquellas que diría un funcionario cuando asiste a gente que quiere renovar el pasaporte y/o el Documento Nacional de Identidad, o información relacionada. En este contexto concreto, un sistema de traducción de voz a LSE es muy útil puesto que la mayoría de estos empleados no conocen este lenguaje y tienen dificultades a la hora de interactuar con personas sordas.

El conjunto de frases se dividió aleatoriamente en tres grupos: entrenamiento (conteniendo aproximadamente el 70% de las frases), evaluación (con el 15% de las frases) y test (15% de frases). Esta concentración se hace de forma arbitraria. Se muestra a continuación un resumen de la base de datos:

		Castellano	**LSE**
Total	**Pares de Frases**	**414**	
	Nº de palabras/glosas	**4847**	**4564**
Entrenamiento	Pares de Frases	314	
	Nº de palabras/glosas	3071	2876
Validación	Pares de Frases	50	
	Nº de palabras/glosas	582	574
Test	Pares de Frases	50	
	Nº de palabras/glosas	570	505

Tabla 7: Estadísticas de la base de datos

6.3.3.3 Resultados de los experimentos realizados

Las 414 frases fueron pronunciadas por 14 personas para evaluar el reconocedor de voz. En este caso se ha realizado tres experimentos diferenciados que se describen a continuación:

- En la primera situación, se evalúa el sistema de reconocimiento de voz utilizando tanto el modelo de lenguaje como el vocabulario generados a partir del conjunto de entrenamiento. Esta situación es la más realista.
- En el segundo caso, el modelo de lenguaje se genera a partir del conjunto de entrenamiento, mientras que el vocabulario incluye todas las palabras. De esta forma se evita el efecto de las palabras fuera de vocabulario.
- En el último experimento se utilizan todas las frases tanto para el entrenamiento como para el vocabulario. En este caso, se intenta reproducir la situación en la que se disponga de tantas frases de entrenamiento que las frases de test estén contenidas en ellas.

A continuación se expresan en forma de tabla los resultados obtenidos para los tres experimentos. Como parámetros de medida se incluyen: WER (Word Error Rate), I (inserciones), D (borrados) y S (sustituciones):

	WER	**I(%)**	**D(%)**	**S(%)**
Experimento 1	24,08	2,61	6,71	14,76
Experimento 2	15,84	1,19	5,93	8,72
Experimento 3	4,74	0,86	1,94	1,94

Tabla 8: Resultados del Reconocedor de Voz para los tres experimentos realizados

A continuación se muestran los resultados de traducción obtenidos aplicando las técnicas de traducción estadística descritas en los apartados 3 y 4. En la Tabla 9 se observan los resultados de los experimentos de traducción realizados, tanto con las frases de referencia del corpus paralelo castellano-LSE (Referencia), como utilizando la salida del reconocedor de voz para los tres experimentos de reconocimiento comentados anteriormente (Experimento 1-3). Por otro lado se diferencian dos situaciones principales: en la primera parte de la tabla se muestran los resultados habiendo entrenado el modelo de traducción con las frases de referencia, en segundo lugar, la segunda parte de la tabla muestra los mismos resultados pero en este caso considerando la salida de reconocedor (de las frases de entrenamiento) para entrenar el modelo de traducción. Para todos los casos se muestran los resultados de WER (tasa de error de signos a la salida de la traducción), tasas de signos insertados, borrados o sustituidos en la traducción y las medidas de BLEU y NIST descritas anteriormente.

Modelo de traducción generado con las frases de referencia del conjunto de entrenamiento				
		WER	BLEU	NIST
Traducción basada en subfrases	Exp 1	39,17	0,4853	6,2806
	Exp 2	37,99	0,4900	6,4006
	Exp 3	33,72	0,5301	6,7238
	REF	31,75	0,5469	6,8652
Traducción basada en FST	Exp 1	35,85	0,5090	6,6473
	Exp 2	33,89	0,5238	6,8610
	Exp 3	**29,32**	**0,5804**	**7,3100**
	REF	**28,21**	**0,5905**	**7,3501**
Modelo de traducción generado con la salida del reconocedor para las frases de entrenamiento				
		WER	BLEU	NIST
Traducción	Exp 1	40,04	0,4775	6,2076

basada en subfrases	Exp 2	37,46	0,4939	6,4738
	Exp 3	32,44	0,5449	6,8606
	REF	31,75	0,5469	6,8652
Traducción basada en FST	Exp 1	36,33	0,5188	6,5273
	Exp 2	33,42	0,5235	6,8344
	Exp 3	**29,27**	**0,5698**	**7,1953**
	REF	**28,21**	**0,5905**	**7,3501**

Tabla 9: Resultado de los experimentos de traducción

En esta tabla se puede observar que los resultados de la Referencia siempre serán los mejores resultados (menor WER y mayor BLEU y NIST) en comparación con los obtenidos en la traducción de la salida del reconocedor de voz puesto que la referencia no contiene errores de reconocimiento que dificultan la traducción posterior. Además, se puede ver que cuanto peor es la tasa de reconocimiento, peor es la tasa de traducción que se consigue traduciendo la salida del reconocedor. En general, con esta base de datos (del dominio de frases del DNI/pasaporte), la traducción estadística basada en FST ofrece mejores resultados que la solución tecnológica basada en subfrases. Se observa también que al entrenar el modelo de traducción con las salidas de reconocimiento se permite entrenar dicho modelo con los posibles errores del reconocedor, de forma que el modelo de traducción puede aprender de estos errores y corregirlos durante el proceso de traducción. Si bien es cierto que los resultados mejoran, las diferencias son muy pequeñas. Finalmente se puede concluir que el mejor sistema es el de la traducción basada en FST entrenando con las salidas del reconocedor.

6.3.3.4 Conclusiones

En este artículo se ha presentado un sistema de traducción estadística de voz a lengua de signos para personas sordas. En concreto se ha estudiado la traducción de castellano a Lengua de Signos Española, utilizando como dominio de aplicación el de frases que un policía pronuncia cuando informa sobre cómo renovar o solicitar el DNI. Estas situaciones requieren de un intérprete signante que sea capaz de traducir cualquier frase a personas sordas que quieran realizar estas acciones por lo que un sistema automático puede ser de gran ayuda. En cuanto al sistema desarrollado, contiene un primer módulo de reconocimiento de voz, un módulo de traducción en el que se ha centrado este artículo, y un último módulo de representación de los signos. Se han estudiado dos soluciones tecnológicas de traducción estadística, la primera utiliza un modelo de traducción basado en subsecuencias de palabras y la segunda utiliza un transductor de estados finitos (FSTs). En ambos casos se utilizan programas de libre distribución que se comentan a lo largo del texto (GIZA++, Moses y GIATI). Estos programas permiten hacer la traducción tanto de textos en castellano (que contienen las frases originales) como de textos

que contienen las frases obtenidas a la salida del reconocedor de voz (que contendrán los fallos propios del reconocimiento) a LSE.

Los resultados que se muestran corresponden a pruebas con el texto original (de referencia) y el texto obtenido a la salida del reconocedor. Para este estudio, la base de datos de frases se ha dividido en tres subconjuntos: entrenamiento, validación y test (comprobación). En general, con esta base de datos (del dominio de frases del DNI/pasaporte), la traducción estadística basada en FST ofrece mejores resultados que la solución tecnológica basada en subsecuencias de palabras. El hecho de entrenar el modelo de traducción con las salidas de reconocimiento permite aprender de los errores y corregirlos durante el proceso de traducción. Si bien es cierto que los resultados mejoran, las diferencias son muy pequeñas. Finalmente se puede concluir que el mejor sistema es el de la traducción basada en FST entrenando con las salidas del reconocedor con una WER de 29,27% y un BLEU de 0,5698.

6.4. Combinación de las diferentes estrategias

La combinación de las tres estrategias de traducción se ha realizado de forma jerárquica siguiendo los siguientes pasos.

1.- En primer lugar se realiza una traducción basada en ejemplos. En el caso de que la distancia entre la frase a traducir y alguno de los ejemplos sea menor que un umbral, la salida del módulo de traducción será la propia traducción del ejemplo más parecido a la frase de entrada.

2.- En el caso de que no haya ningún ejemplo suficientemente parecido a la frase a traducir se procede a ejecutar la traducción basada en reglas. Posteriormente se analiza la salida de este tipo de traducción y se calcula el número de signos/glosas generadas en relación con el número de palabras de la frase a traducir. Generalmente, si la traducción basada en reglas tiene problemas, ese hecho se manifiesta porque se generan muy pocos signos en relación con las palabras de la frase de entrada. Es decir, se ejecutan pocas reglas que generen signos.

3.- En el caso en el que no se generen suficiente número de signos se recurre a la traducción estadística para generar el resultado de la traducción.

Tanto el umbral de distancia como el umbral de tasa de signos/palabras se han ajustado manualmente en función del dominio de aplicación definido.

7. Módulo de representación de los signos

Para la representación de los signos se ha empleado un agente virtual animado en 3D, el avatar "virtual Guido" (VGuido), desarrollado por Televirtual Ltd. en el proyecto eSign ("Essential Sign Language Information on Government Networks")2. Un avatar genera una animación mediante una sucesión temporal de imágenes, cada una de las cuales representa una postura estática del avatar, que a su vez queda definida especificando la configuración del esqueleto del agente animado.

Figura 27: Agente virtual VGuido

En el proyecto ViSiCAST ("Virtual Signing: Capture, Animation, Storage and Transmission"), predecesor de eSign, los signos se generaban originalmente mediante captura de movimiento. Esta técnica consiste en que un experto en Lengua de Signos, equipado con un sistema que permite captar sus movimientos, representa los signos uno por uno. Éstos quedan registrados en un fichero, permitiendo con ello animar agentes virtuales directamente. Sin embargo, esta técnica tiene serios inconvenientes. En primer lugar, el equipo es muy caro, y por tanto, no es fácil disponer de él, de forma que una vez que un signo es almacenado, no es fácil modificarlo, no pudiendo adaptarse en caso de una variación lingüística. Además, la configuración y calibración del equipo para cada sesión son tareas difíciles de realizar y que consumen mucho tiempo. Por último, el equipo es incómodo de llevar, y además no es capaz de detectar movimientos pequeños, por lo que muchos signos deben ser modificados después de su captura.

Es por estas razones que VGuido no emplea captura de movimiento, sino que la animación es generada a partir de descripciones de los signos mediante HamNoSys ("Hamburg Notation System" – "Sistema de Notación de Hamburgo"), un sistema de notación para la Lengua de Signos, independiente

[2] **Proyecto eSign** (*"Essential Sign Language Information on Government Networks"* – "Información Esencial en Lengua de Signos en Redes Gubernamentales"). Proyecto europeo creado como respuesta a la necesidad de generar contenido en Lengua de Signos de una forma eficiente. Se crea a partir del proyecto europeo ViSiCAST (*"Virtual Signing: Capture, Animation, Storage and Trasmission"* – "Signado Virtual: Captura, Animación, Almacenamiento y Transmisión"), desarrollado por compañías y universidades europeas, en Francia, Alemania, Gran Bretaña, y los Países Bajos.

del avatar. Mediante HamNoSys se definen aspectos relacionados con la posición de las manos, la velocidad de signado, la amplitud del gesto, etc.

HamNoSys se centra en la parte manual de los signos. Sin embargo, la implementación de la parte no manual (movimientos de cabeza, ojos, boca, etc.) también es fundamental en la Lengua de Signos. En el capítulo siguiente describiremos el editor de signos desarrollado que permite transcribir los signos de una forma rápida y sencilla, incorporando además las herramientas necesarias para describir la parte no manual de los signos.

Dado que los ordenadores no pueden procesar la sintaxis de estas descripciones en HamNoSys, en la Universidad West Anglian se diseñó SiGML ("Signing Gesture Markup Language" – "Lenguaje Marcado de Gestos Signados"), una versión modificada de XML3, y se generó un traductor de HamNoSys a SiGML. La descripción en SiGML de un signo contiene exactamente la misma información que la descripción en HamNoSys, pero en un lenguaje que pueden procesar los ordenadores. Esta información es procesada en un sintetizador, que, teniendo en cuenta la descripción de la geometría del agente animado, completa todos los detalles que la transcripción SiGML pueda haber omitido, como la localización por defecto del avatar, la duración en segundos de cada movimiento (expresadas en SiGML simplemente como velocidad rápida, lenta o normal), etc., dando lugar a una transcripción SiGML más detallada, SiGML Extendido, que puede describir algunos detalles que no es posible implementar con HamNoSys.

Finalmente, esta descripción de los signos es representada mediante el agente animado virtual. En la figura siguiente se muestra el diagrama de bloques del proceso seguido para generar una animación de un signo.

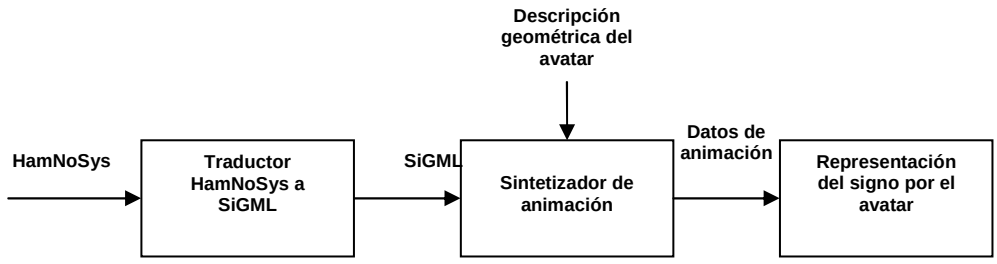


Figura 28: Diagrama de bloques de la generación de un signo animado

Los sofisticados modelos de movimiento empleados permiten tener una buena calidad en la representación, en términos de reconocimiento de signos, próxima a la calidad en la representación generada a partir de captura de movimientos. La principal ventaja de esta técnica es su flexibilidad, ya que la generación del vocabulario de signos no requiere ningún equipo especial, y la

[3] **XML** (*"eXtensible Markup Language"* – "Lenguaje Marcado Extensible"). Metalenguaje extensible de etiquetas, desarrollado por el World Wide Web Consortium, que permite definir la gramática de lenguajes específicos. Por lo tanto, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Permite compatibilidad entre sistemas para compartir la información de una manera segura, fácil y fiable.

descripción de los signos simplemente se almacena en una base de datos. De esta forma pueden formarse rápidamente frases en la Lengua de Signos, seleccionando los signos necesarios de la base de datos y concatenándolos en el orden que corresponda.

7.1. Instalación del agente animado

Para la representación animada de los signos es necesario instalar el software correspondiente: el agente animado VGuido y los módulos de animación se han incorporado en el sistema como un control ActiveX.

Para poder instalar el avatar VGuido y visualizarlo correctamente, el sistema debe cumplir los siguientes requisitos:

- Procesador: Windows Intel Pentium
- Sistema Operativo: Microsoft Windows 2000 con Service Pack 2, o Windows XP Professional o Home Edition.
- Memoria: 128 MB de RAM. 30 MB de espacio disponible en disco.
- Tarjeta gráfica: nVidia GeForce4 o de calidad equivalente.
- Navegador: Internet Explorer 6.
- Java: Entorno Java (JRE, *"Java Runtime Environment"*) 1.4 o superior. El programa de instalación comprueba automáticamente si el JRE ya está instalado en el PC.

Actualmente, el avatar sólo está disponible para Windows, no hay una instalación disponible para ordenadores Mac o para otros navegadores distintos de Internet Explorer.

El avatar VGuido puede descargarse libremente para uso personal desde la página web de eSign, http://www.sign-lang.uni-hamburg.de/eSIGN/Software.html, en el paquete eSign Avatar Plugin Installer. Este software permite visualizar el agente animado VGuido en aquellas páginas web que incorporen el avatar.

Sin embargo, si se quiere incorporar el agente animado en una aplicación de Windows, como es el caso de este proyecto, es necesario instalar la versión 2.6.7 del control ActiveX SiGML Signing. Este control ActiveX es incompatible con el Plugin anterior, por lo que para un correcto funcionamiento del control ActiveX debe desinstalarse el Plugin previamente. Al instalar el paquete SiGML Signing 2.6.7 se instalan una serie de aplicaciones para la representación de signos con el avatar, una serie de ejemplos, y un conjunto de componentes, entre los cuales cabe destacar el SiGMLSigning.dll. Éste es el componente principal del paquete, y utiliza otros componentes instalados para proporcionar la funcionalidad que necesitan las aplicaciones de signado. Permite convertir SiGML a animación. Tras la instalación del paquete se generará la carpeta SiGMLSigning en C:\Archivos de Programa\eSign.

El software incluye toda la información relevante acerca de las características del avatar:

- Posiciones de las articulaciones del agente animado.
- Información sobre puntos característicos que se pueden describir con el sistema HamNoSys, como “el centro de la frente”, “la punta de la barbilla”, etc.
- Tipo de cada articulación: si funciona como una bisagra, si encaja una parte con otra, etc.
- Información sobre aspectos del lenguaje corporal: velocidad de generación de signos, cómo de rápidas son las paradas si una mano se mueve a un punto determinado, el tiempo típico que tarda una mano en moverse de una posición a otra, etc.

Para una descripción más detallada de las aplicaciones y componentes se puede consultar el ReadMe del software instalado.

7.2. Introducción al sistema de notación de HamNoSys

El Sistema de Notación de Hamburgo (HamNoSys, “Hamburg Notation System”), es un sistema de transcripción fonética para la Lengua de Signos, que comprende más de 200 símbolos. Fue desarrollado en los años 1980 en el Instituto de Lengua de Signos Alemana de la Universidad de Hamburgo, para la transcripción tanto de signos individuales como de frases en la Lengua de Signos.

Hasta el momento, HamNoSys se ha empleado como un sistema de notación de referencia en varios proyectos de investigación en Lengua de Signos. Sin embargo, en el contexto de generación signos, ViSiCAST es el primer proyecto que ha empleado HamNoSys para almacenar en el vocabulario la transcripción fonética de signos individuales, y combinarlos posteriormente para formar frases. Al implementar ViSiCAST se decidió emplear HamNoSys por varias razones. En primer lugar, es el sistema de notación más estable y más frecuentemente empleado entre la comunidad de Lengua de Signos. Además, es un sistema independiente de la lengua, es decir, puede aplicarse indistintamente para todas las lenguas de signos. Por último, las notaciones son compactas y sencillas de introducir en un entorno de edición de signos.

HamNoSys se ha diseñado siguiendo los principios básicos que se describen a continuación:

- Independencia del lenguaje. HamNoSys no es específico para una Lengua de Signos determinada, sino que permite describir cualquier signo de cualquier lengua. Esta característica es consecuencia de la motivación original que llevó a la implementación de HamNoSys: proporcionar un medio escrito a los investigadores para describir signos.
- Descripción de postura y movimiento, no de significado. El significado de un signo no queda descrito en HamNoSys, sólo se describen la postura y movimiento de las manos. Un signo puede tener distintos significados en distintos contextos, pero si se representa de la misma manera, la transcripción HamNoSys es la misma.

- Omisión de información irrelevante. Sólo se describen las partes de la postura y movimiento que son importantes para crear un signo. La mayoría de los signos se representan a través de las manos y la cara, de forma que la posición de, por ejemplo, los hombros y codos no es importante, simplemente deben adaptarse de forma natural a la posición de las manos. Así, los hombros y los codos no tienen notación en la mayoría de los signos.

HamNoSys permite describir los signos en términos de forma, orientación, posición, y movimiento de las manos. Cada uno de estos aspectos se describe a continuación en los siguientes apartados.

7.2.1. Forma de la mano

En el sistema HamNoSys se definen 12 formas estándar de la mano: mano abierta, mano cerrada, mano en forma de C, dedo índice extendido, dos dedos extendidos (juntos o separados), etc.

Sobre estas formas se puede aplicar un conjunto de modificaciones, cambiando la inclinación de los dedos o la posición del dedo pulgar. A continuación se muestran algunos ejemplos.

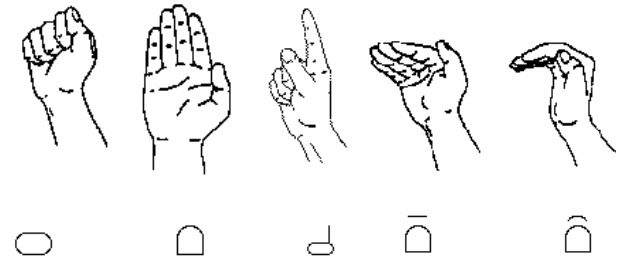


Figura 29: Transcripción HamNoSys para algunas formas de la mano.

En la figura se muestra la representación HamNoSys para distintas formas básicas de la mano, así como algunas de las modificaciones de la inclinación de los dedos que se pueden realizar.

En cada caso se pueden indicar los dedos a los que se refiere la transcripción. Así por ejemplo, el símbolo "☝" se refiere por defecto a la forma de la mano cerrada con el dedo índice extendido. Sin embargo, en la transcripción se puede indicar el dedo que tiene que estar extendido, simplemente añadiendo su símbolo correspondiente (número del 1 al 5) a continuación, tal y como se observa en la siguiente representación con VGuido, donde el dedo extendido es el meñique.

Figura 30: Transcripción HamNoSys para la forma de la mano con el dedo meñique extendido

El sistema también admite descripciones para cruzar los dedos, o esconder el dedo pulgar entre dos dedos determinados. Por ejemplo, mediante la expresión  , se cruza el tercer dedo sobre el dedo índice.

Además, se pueden transcribir formas intermedias entre dos cualquiera de las definidas, mediante el símbolo \ situado entre dos símbolos HamNoSys. En la siguiente figura se muestra un ejemplo de esta situación.

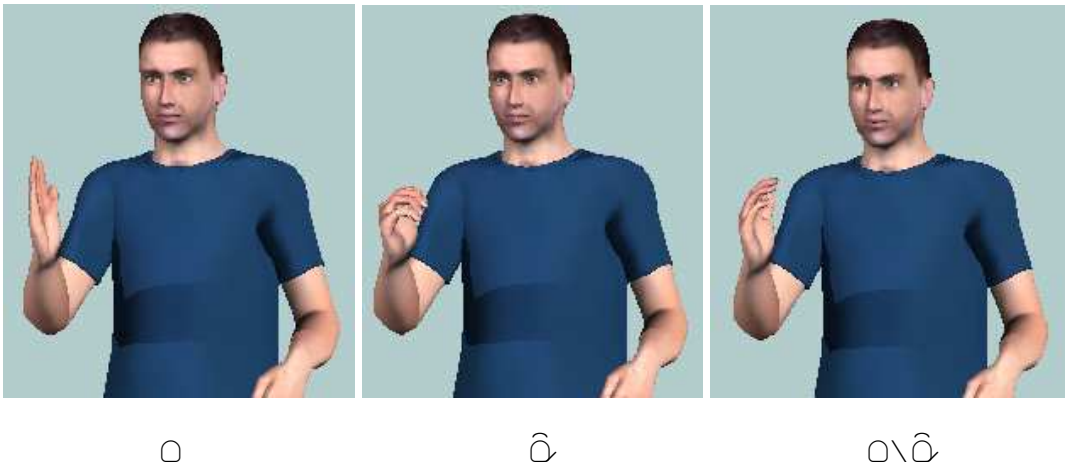


Figura 31: Transcripción HamNoSys para dos formas de la mano y su posición intermedia

7.2.2. Orientación de la mano

La información de orientación tridimensional de la mano está dividida en dos parámetros: la dirección de la base del dedo índice y la orientación de la palma de la mano. El primero define la dirección del eje de la mano, mientras que el segundo define la orientación de la mano según el eje.

La dirección de la base del dedo índice es la dirección a la que estaría apuntando el dedo índice si estuviera extendido. Tiene 26 valores posibles, que corresponden a las direcciones que se pueden trazar desde el centro de un

cubo a los centros de sus caras, a los puntos medios de sus bordes, y a sus vértices.

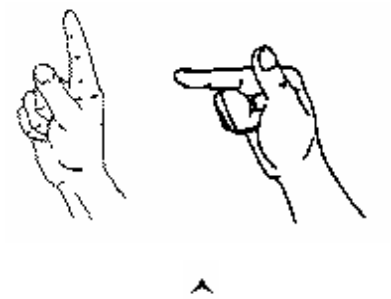


Figura 32: Trascripción HamNoSys para la dirección de orientación hacia arriba.

En la figura se muestra un ejemplo de dirección del eje de la mano, con su transcripción en HamNoSys. Ambas representaciones tienen la misma dirección de la base del dedo índice, cambiando únicamente la inclinación del dedo.

En la figura siguiente se muestran algunas de las posibles direcciones del eje de la mano, con sus transcripciones en HamNoSys.

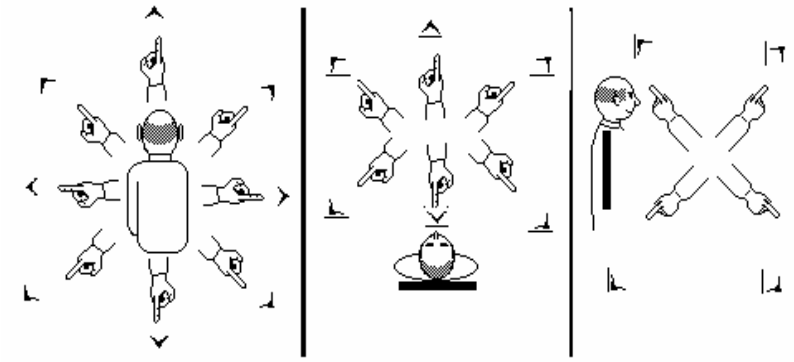


Figura 33: Trascripción HamNoSys de algunas posibles direcciones de la base del dedo índice.

Además de las indicadas en la figura, se pueden definir otras ocho direcciones intermedias entre éstas, introduciendo un símbolo de dirección de la base del dedo índice a continuación de otro, como se indica a continuación:

La orientación de la palma puede tomar ocho valores distintos, cada uno de ellos indicando una orientación de la palma alrededor del eje de la mano. Es decir, si por ejemplo la dirección de la base del dedo índice es hacia delante, la palma puede estar hacia arriba, hacia abajo, hacia la derecha, hacia la izquierda, o hacia cualquiera de las cuatro orientaciones intermedias entre éstas.

A continuación se presentan algunos ejemplos de dirección del eje de la mano y orientación de la palma según este eje.

: Dirección del eje de la mano hacia delante, palma hacia abajo.

: Dirección del eje de la mano hacia arriba, palma de frente.

: Dirección del eje de la mano hacia arriba, palma hacia la izquierda.

La parte oscura del símbolo HamNoSys representa siempre la palma de la mano.

En algunos signos, la orientación de la mano varía constantemente durante un movimiento (por ejemplo, en los movimientos ondulatorios), conservando una relación constante con él. Para transcribir esta situación se emplea el símbolo ~, que se introduce a continuación del símbolo de orientación.

Tanto para la dirección de la base del dedo índice como para la orientación de la palma de la mano se pueden transcribir orientaciones intermedias entre dos cualquiera de las definidas, mediante el símbolo \ situado entre dos símbolos HamNoSys. A modo de ejemplo, en la siguiente figura se muestra la dirección intermedia entre la dirección horizontal hacia la izquierda y la vertical hacia arriba. El resultado es equivalente al obtenido al introducir directamente el símbolo HamNoSys .

Figura 34: Transcripción HamNoSys para dos direcciones del eje de la mano y posición intermedia

7.2.3. Localización de la mano

La mano se puede situar en distintas partes del cuerpo, y a distintas distancias: en contacto con el cuerpo (), cerca (), a una distancia normal (por defecto), o lejos del cuerpo (). Existen decenas de símbolos en HamNoSys para transcribir la localización de la mano: cabeza (), hombros (), pecho (), nariz (), barbilla (), palma de la mano contraria (), etc., y la mano puede situarse a la derecha, a la izquierda, o en el centro de cada una de estas partes del cuerpo. En la siguiente tabla se muestran algunas de las localizaciones más frecuentemente utilizadas.

	A la izquierda	En la parte izquierda	Centro	En la parte derecha	A la derecha
Cabeza	▫◯	▪◯	◯	◯▪	◯▫
Frente	▫⊓	▪⊓	⊓	⊓▪	⊓▫
Nariz	▫ ⫛	▪ ⫛	⫛	⫛ ▪	⫛ ▫
Boca	▫⬭	▪⬭	⬭	⬭▪	⬭▫
Barbilla	▫◡	▪◡	◡	◡▪	◡▫
Cuello	▫⟩⟨	▪⟩⟨	⟩⟨	⟩⟨▪	⟩⟨▫
Línea de los hombros	▫▽	▪▽	▽	▽▪	▽▫
Línea del pecho	▫▽	▪▽	▽	▽▪	▽▫
Línea del estómago	▫▽	▪▽	▽	▽▪	▽▫

Tabla 10: Ejemplo de algunas de las localizaciones más frecuentemente utilizadas

Cuando hay contacto entre la mano y alguna parte del cuerpo, se puede indicar entre corchetes con qué parte de la mano exactamente se realiza el contacto (punta de un dedo, palma/ dorso de la mano, parte lateral derecha/ izquierda de un dedo, etc.). En el siguiente ejemplo se muestra la representación con VGuido para el signo “ALEMANIA”, y su transcripción HamNoSys, donde se indica que se debe tocar la frente con el dedo pulgar.

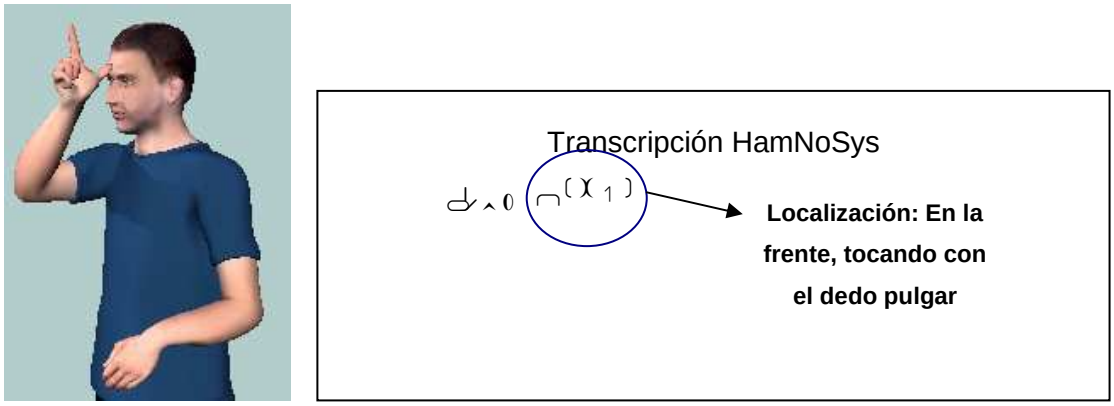


Figura 35: Representación mediante VGuido del signo “ALEMANIA” y su transcripción HamNoSys

Es posible transcribir cualquier posición intermedia entre dos cualquiera de las definidas, mediante el símbolo \ situado entre dos símbolos HamNoSys. En la siguiente figura se muestra un ejemplo de la situación descrita, en el que se obtiene la posición intermedia entre la localización de la mano a la altura del pecho y la localización de la mano a la derecha de la cabeza.

Figura 36: Trascripción HamNoSys para dos localizaciones y su posición intermedia

7.2.4. Movimientos de la mano

La descripción de los movimientos puede llegar a ser bastante compleja. Se pueden definir movimientos de las manos en el espacio, o bien movimientos de las manos sin cambiar su localización (por ejemplo, flexión de los dedos).

Los distintos movimientos que puede realizar la mano a través de espacio pueden ser:

- **Rectos.** La descripción de los movimientos en línea recta sigue los mismos principios que la descripción de la dirección de la base del dedo índice, de forma que pueden realizarse movimientos en las 26 direcciones distintas descritas anteriormente para el eje de la mano. Por ejemplo, la trascripción en HamNoSys para un movimiento en línea recta hacia arriba corresponde a ↑, y hacia delante a ↥.
- **Curvos.** Tras indicar la dirección del movimiento mediante un símbolo de descripción de movimiento en línea recta, se introduce uno de los símbolos de movimiento curvo, donde el plano de la curva puede orientarse de 8 formas diferentes, de forma similar a las distintas orientaciones posibles de la palma de la mano. Por ejemplo, un movimiento hacia arriba puede hacer una curva hacia la derecha, hacia la izquierda, hacia fuera y hacia dentro, como se muestra en la siguiente figura.

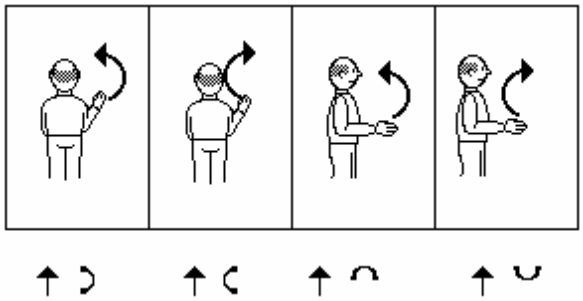


Figura 37: Transcripción HamNoSys de algunos ejemplos de movimiento curvo.

Además se pueden definir otros cuatro movimientos intermedios entre éstos, introduciendo un símbolo de movimiento curvo a continuación de otro, como se muestra a continuación:

- **Circular.** Se pueden definir tres planos de representación del movimiento circular: horizontal (,), vertical en el plano paralelo a la persona (,), y vertical en el plano perpendicular a la persona (,). Además, pueden definirse los planos intermedios entre ellos. Para cada uno de los movimientos circulares descritos se puede definir un sentido de giro o el contrario. Es posible también establecer el punto del círculo en que comienza el movimiento y el punto en que termina, mediante los símbolos siguientes: . A continuación se muestran algunos ejemplos de uso de estos símbolos.

Ejemplo	
:	Movimiento circular en el sentido de las agujas del reloj, que empieza y termina en la parte superior.
:	Movimiento de tres cuartos de circunferencia en el sentido de las agujas del reloj, que empieza en la parte superior.
:	Movimiento de siete cuartos de circunferencia en el sentido de las agujas del reloj, que empieza en la parte superior.

Pueden describirse además elipses, introduciendo los símbolos correspondientes (, , ,) a continuación del símbolo de movimiento circular. Así por ejemplo la expresión correspondería a una elipse representada en frente del cuerpo, más ancha que alta.

- **Dirigido a una localización en particular.** Mediante el símbolo es posible cambiar la localización de la mano, situándola en el lugar que interese.

También es posible definir uno de los tres tipos de movimientos descritos anteriormente, y añadir a continuación la transcripción HamNoSys de la localización donde debe finalizar el movimiento.

Para cada uno de los movimientos indicados (excepto en el caso de que el movimiento esté dirigido a una localización en particular y se utilice el símbolo ), es posible definir un movimiento en zigzag o un movimiento ondulatorio ( , ). Además, puede establecerse la longitud del movimiento en línea recta o curva, y el tamaño del diámetro del movimiento circular. Concretamente, un movimiento corto o de diámetro pequeño se indica mediante el símbolo  , y un movimiento largo o de diámetro grande, mediante el símbolo  . Por último, puede indicarse también si el movimiento es rápido, lento, si termina bruscamente, etc.

Es posible repetir un movimiento, una () o varias veces (), y en sentido contrario al movimiento realizado (), o en el mismo sentido. Además, se puede indicar que las repeticiones de los movimientos vayan disminuyendo o aumentando en amplitud, o que empiecen en el punto en que terminó el anterior o vuelvan al mismo punto de origen. Por ejemplo, la expresión  genera un movimiento hacia abajo, que se repite varias veces, cada vez con una amplitud más pequeña. De igual manera, la expresión  también genera varios movimientos hacia abajo, cada vez con menor amplitud, pero en este caso cada movimiento empieza en el punto en que terminó el anterior.

En cuanto a los ***movimientos de las manos sin cambiar la localización***, pueden referirse a:

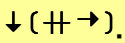
- **Cambios en la forma u orientación de la mano.** El cambio de forma o de orientación de la mano (ya sea de la dirección de la base del dedo índice o de la orientación de la palma), se indica mediante el símbolo , introduciendo en el lado izquierdo de la flecha la forma/ orientación original, y en el lado derecho la forma/ orientación final.
- **Movimiento de muñeca.** La muñeca puede moverse repetidas veces en vertical (arriba – abajo), mediante el símbolo , y en horizontal (derecha – izquierda), mediante el símbolo . Además, admite movimientos de rotación.
- **Movimiento de dedos.** Mediante el símbolo se puede describir un movimiento de los dedos denominado *"fingerplay"* (juego de dedos).

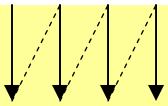
Todos los movimientos descritos pueden combinarse secuencialmente o en paralelo. Es decir, puede realizarse un movimiento a continuación de otro, simplemente introduciendo en el orden correspondiente los símbolos HamNoSys de los movimientos que se desean realizar, o bien repetir un movimiento mientras se realiza otro. A continuación se muestran algunos ejemplos de esta última situación. Es importante destacar el uso de los paréntesis  y los símbolos de repetición para poder implementar correctamente estos movimientos.

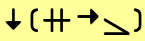
Ejemplo. Movimiento de la muñeca durante un movimiento lineal o circular.

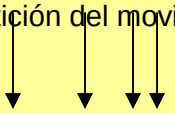
 : Rotación continua de la muñeca hacia la derecha mientras se dibuja un círculo en el espacio.

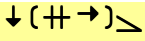
Ejemplo. Repetición de un movimiento mientras la mano se desplaza.

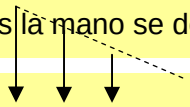
 : Movimiento hacia abajo que se repite varias veces mientras la mano se desplaza hacia la derecha.



 : Movimiento hacia abajo que se repite varias veces mientras la mano se desplaza hacia la derecha, habiendo cada vez una separación menor entre cada repetición del movimiento.



 : Movimiento hacia abajo que se repite varias veces, con menor amplitud cada vez, mientras la mano se desplaza hacia la derecha.



7.2.5. Representación de signos con dos manos

De la misma forma en que se han definido los parámetros necesarios para una sola mano, la mano dominante, pueden definirse para la mano no-dominante. Las dos manos pueden tener la misma o distinta forma, orientación, localización, y movimiento, y estos parámetros pueden ser simétricos respecto al eje horizontal (**:**) o respecto al eje vertical (¨). Además, puede definirse la distancia entre las dos manos, describiendo si están en contacto (), cerca (), lejos (), o a una distancia normal. Cabe señalar la diferencia en la interpretación de la transcripción HamNoSys dependiendo de si estos símbolos de distancia se encuentran antes o después de un símbolo de localización. Este hecho puede explicarse fácilmente mediante el siguiente ejemplo. En una descripción HamNoSys con dos manos, la expresión  indica que las manos

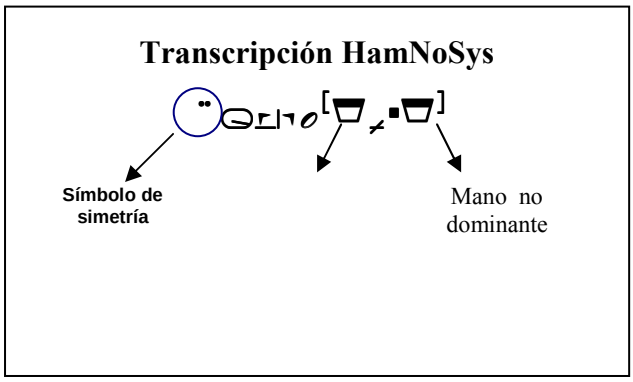
se encuentran a la altura del pecho, a una distancia normal del cuerpo y en contacto entre ellas. Sin embargo, la expresión  indica que las manos están en contacto con el cuerpo a la altura del pecho, y no hacen contacto entre ellas.

Si ambas manos tienen la misma forma, orientación y localización, y se mueven de la misma manera, para transcribir el signo es suficiente con poner el símbolo de simetría, y a continuación la descripción del signo como si fuera para una sola mano. En este caso, las manos se sitúan cerca entre ellas, de forma que si se quiere que estén alejadas, hay que indicarlo expresamente, describiendo la localización para cada mano. Si, por el contrario, alguno de los parámetros entre las dos manos es distinto, la transcripción HamNoSys tiene la siguiente estructura:

Símbolo_de_simetría Parámetros_comunes_entre_las_dos_manos [Parámetros mano dominante , Parámetros mano no dominante]

En la siguiente figura se representa el signo “AUTORIDAD”, que utiliza dos manos, junto con su transcripción HamNoSys.

Figura 38: Representación del signo “AUTORIDAD” y su transcripción HamNoSys



Para más información acerca del sistema HamNoSys se puede consultar la página web de la Universidad de Hamburgo:

http://www.sign-lang.uni-hamburg.de/Projekte/HamNoSys

y el documento *“Deliverable D5.1: Interface Definitions”*, que se puede descargar desde la página web de ViSiCAST:

http://www.visicast.cmp.uea.ac.uk/Papers/ViSiCAST_D5-1v017rev2.pdf

7.3. Limitaciones observadas en la representación de los signos

Durante la realización de la tarea de generación de los signos se han observado algunas limitaciones del entorno VGuido en la representación de los signos. En este apartado se describen estas limitaciones, y las soluciones utilizadas para poder representar los signos afectados.

El agente animado no está diseñado para soportar algunas de las transcripciones HamNoSys, de forma que las ignora, o bien no realiza ningún movimiento. De igual manera, cuando se transcribe un signo en HamNoSys

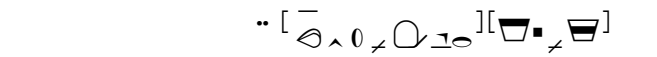
con algún error de notación, se genera un fichero .SiGML que el agente animado no puede representar, de forma que no produce ningún movimiento.

7.3.1. Generación de signos con dos manos

Al generar los signos con dos manos, debe definirse la forma, orientación, localización y movimiento para cada mano. Si el comportamiento de ambas manos es idéntico, la descripción HamNoSys es igual que para los signos con una sola mano, pero precedido de un símbolo de simetría. Sin embargo, si en algún aspecto las manos se comportan de forma diferente, este hecho debe indicarse introduciendo la descripción HamNoSys entre dos corchetes separados por el símbolo . La descripción de la mano dominante se encuentra a la izquierda del símbolo , y la de la mano no dominante, a la derecha.

Para que el agente animado represente el signo correctamente, los símbolos de localización no pueden introducirse en el mismo corchete que los símbolos de forma y orientación, deben introducirse en corchetes distintos.

Por ejemplo, el signo “PONER” se transcribe en HamNoSys con la forma, orientación y localización descritos mediante la siguiente notación:



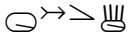
Si los símbolos de localización se introdujeran en el primer corchete, no se podría representar el signo.

7.3.2. Cambio en la forma de la mano: extender o cerrar un dedo tras otro

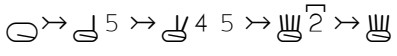
Los símbolos HamNoSys y permiten transcribir aquéllos signos en los que la forma de la mano cambia de mano cerrada a mano abierta o viceversa, extendiendo/doblando los dedos de uno en uno, uno tras otro, comenzando por el meñique o por el pulgar según el símbolo introducido.

Sin embargo, al introducir estos símbolos en la descripción del signo, el agente animado VGuido no sólo no realiza el movimiento descrito, sino que actúa como si la descripción del signo fuera errónea, y no realiza ningún movimiento.

Así, para implementar estos cambios, se ha modificado la forma de la mano por pasos, extendiendo/doblando en cada paso el dedo correspondiente. Por ejemplo, en el signo “ALGUNO” la forma de la mano inicial es cerrada, y los dedos se van extendiendo uno a uno, empezando por el meñique, hasta tener la mano abierta. Lo ideal sería implementarlo de la forma:



Pero dado que esta expresión no es válida para la representación con VGuido, se ha debido implementar por pasos, extendiendo un dedo en cada paso:



La transcripción HamNoSys utilizada comienza con la mano cerrada. A continuación se extiende el dedo meñique, y en el siguiente paso, se describe

la forma de la mano con el dedo meñique y anular extendidos. Después se extienden los dedos meñique, anular y corazón, forma que se transcribe con el símbolo de mano abierta, pero con el dedo índice flexionado. Por último, se describe la forma de la mano completamente abierta. Este proceso se muestra en la siguiente figura.

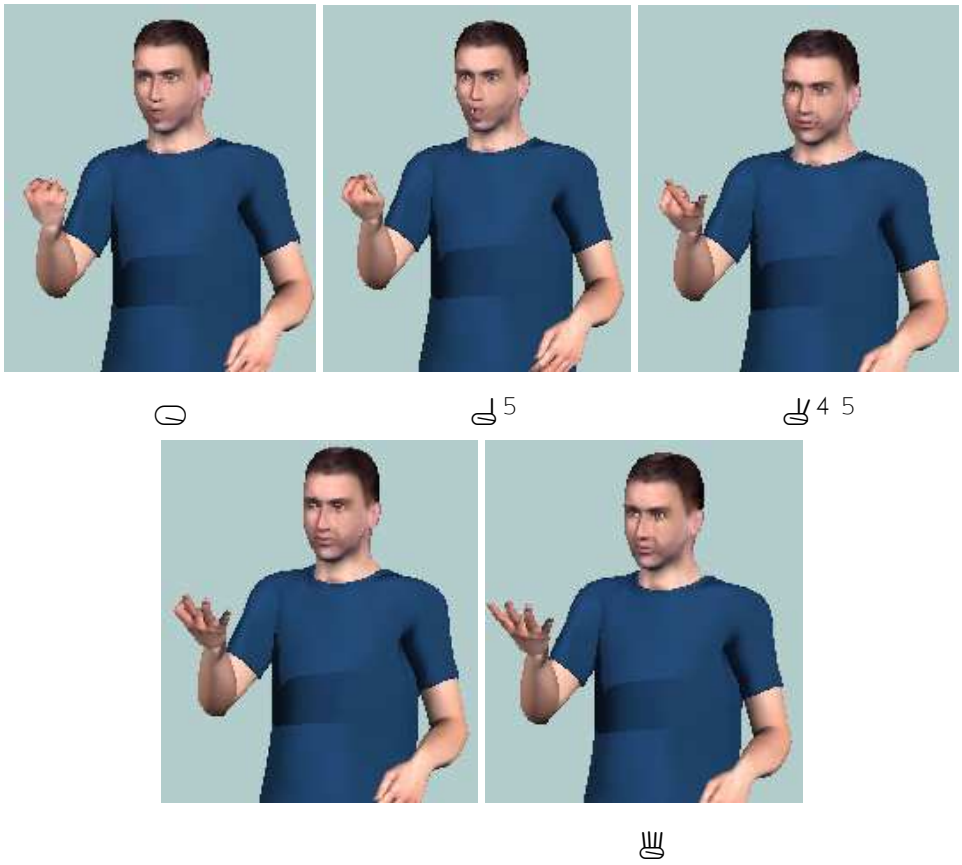


Figura 1. Representación de las transiciones del signo "ALGUNO"

7.3.3. Dedo pulgar entre dos dedos

Con la mano cerrada, el dedo pulgar puede situarse entre dos dedos mediante expresiones del tipo ⌓dedo1\dedo2, donde tras el símbolo que transcribe la forma de la mano cerrada, se indican los dedos entre los que se debe situar el pulgar.

Sin embargo, al introducir esta expresión en la descripción del signo, VGuido la ignora, dejando el pulgar en la situación que se indique en la transcripción de la forma de la mano.

Esta limitación no ha podido solucionarse de ninguna manera, de forma que se ha dejado el pulgar cerrado, opción más adecuada de entre las opciones posibles de posición del dedo pulgar.

7.3.4. Cambio de orientación de la mano durante el movimiento

En algunos signos, la orientación de la mano cambia constantemente con el movimiento, manteniendo una relación constante entre la orientación de la

base del dedo índice/orientación de la palma de la mano y el camino seguido por el movimiento. Esta característica se indica mediante el símbolo: ˷.

Sin embargo, al introducir este símbolo en la descripción, el agente animado lo ignora, actuando como si no estuviera. Así, los signos en los que se produce un cambio de orientación con el movimiento deben implementarse de forma manual, realizando cambios sucesivos de la orientación y localización que permitan representar el signo correctamente.

Por ejemplo, en el signo “ADJUNTAR” la mano derecha describe un movimiento curvo hacia abajo y hacia la izquierda, en el que la orientación de la palma de la mano sigue el movimiento. Podría transcribirse en HamNoSys como:

Sin embargo, esta trascripción no genera el signo correctamente, se realiza el movimiento, pero la orientación de la palma de la mano no cambia con él. Para solucionarlo, se ha descrito el signo por pasos, de la forma:

La representación de cada una de las transiciones en la descripción del signo se muestra en la siguiente figura.

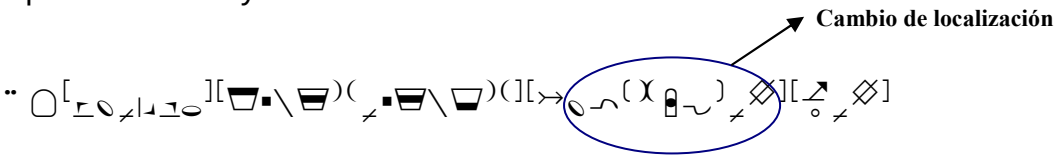
Figura 39: Representación de las transiciones del signo “ADJUNTAR”

7.3.5. Cambio de localización de las manos

Los movimientos dirigidos a una localización en concreto pueden transcribirse mediante el símbolo HamNoSys ↛, introduciendo a la izquierda la localización original, y a la derecha la final.

Si la mano cambia de forma u orientación además de localización, no hay ningún problema en la representación del signo, pero si sólo se modifica la localización, el agente animado no responde. Así, antes de indicar la localización final, siempre hay que indicar la forma o la orientación de la mano, aunque no se realice ninguna modificación sobre ellas.

Por ejemplo, el signo “PARA” se transcribe mediante la siguiente descripción HamNoSys:



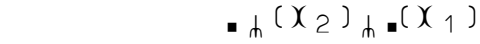
En la parte resaltada se observa que se produce únicamente un cambio de localización de la mano derecha, pero para que el signo se represente correctamente, se ha indicado la orientación de la palma de la mano, aunque no varía con respecto a la inicial.

Figura 40: Cambio de localización de la mano en la representación del signo “PARA”

7.3.6. Localización de los dedos

En algunos signos, la mano se sitúa de forma que un dedo está en contacto con una parte del cuerpo, y otro dedo con otra. Para implementar esta situación, debería simplemente introducirse cada una de las localizaciones de contacto, seguida del dedo con el que hace contacto.

Por ejemplo, un signo en el que el dedo índice tocara la parte izquierda de la nariz y el pulgar la parte derecha, se describiría de la forma:



Sin embargo, cuando esta expresión se introduce en el sistema, el agente VGuido no representa ningún movimiento, tomando la expresión como incorrecta. Así, para representar este tipo de signos, se han debido elegir los demás parámetros (forma de la mano, orientación y localización de la mano) de forma adecuada para que el resultado sea similar al que se quiere representar.

7.3.7. Movimientos de la muñeca

No es posible realizar movimientos de muñeca (arriba-abajo, izquierda-derecha y rotación) a la vez que se está realizando un movimiento en línea recta, curvo o circular.

En los signos en los que la muñeca se mueve a la vez que se produce un desplazamiento de la mano, se ha realizado este desplazamiento por pasos, cambiando la orientación de la palma de la mano y la localización. Por ejemplo, en el signo “BRILLO”, la mano derecha se desplaza hacia la derecha mientras se produce un movimiento de rotación de muñeca (). Como no es posible implementar este tipo de movimiento, se ha modificado la orientación de la palma de la mano en varios pasos, así como la localización, desplazando la mano hacia la derecha en cada paso. En la siguiente se muestran dos pasos del proceso, que se repiten sucesivas veces, cambiando la localización hacia la derecha.

Figura 41: Representación de las transiciones del signo “BRILLO”

7.3.8. Fingerplay

En teoría, el movimiento *“fingerplay”* podría aplicarse solamente a unos dedos concretos. Por ejemplo, en signos con formas de la mano del tipo ∋, debería ser posible mover únicamente los dedos extendidos mediante la expresión ∋(3 4 5).

Sin embargo, al introducir este tipo de expresiones en el sistema, el agente animado VGuido no representa ningún movimiento, tomando la expresión como incorrecta. Dado que este tipo de movimiento no puede implementarse por pasos, como en los casos anteriores, se ha debido prescindir de él, o bien introducirlo para todos los dedos, tomando la opción más adecuada según el signo a representar.

7.3.9. Movimientos de cabeza y cuerpo.

El agente animado no es capaz de representar los movimientos de cabeza y cuerpo, que se definen en la ventana “Limbs”. A pesar de que la traducción de los símbolos en HamNoSys a SiGML se realiza correctamente, VGuido ignora estos símbolos, representando el signo como si no estuvieran.

Así, se ha debido prescindir de estos movimientos, que, por otro lado, son menos relevantes para la representación de los signos que otros movimientos no manuales, como la expresión facial o los gestos de la boca.

7.4. Uso del Control ActiveX para la representación de los signos

Para poder representar los signos de una forma rápida y sencilla, se ha implementado el programa “ejemplo” en Visual C++. Al ejecutarse, genera una ventana donde se introduce directamente el texto SiGML con la descripción HamNoSys correspondiente al signo a representar, que se visualiza mediante el agente animado VGuido. Este ejemplo ha sido integrado en la arquitectura de traducción descrita en el tema 4. Veamos la descripción de este ejemplo para entender el funcionamiento y las posibilidades del agente animado utilizado.

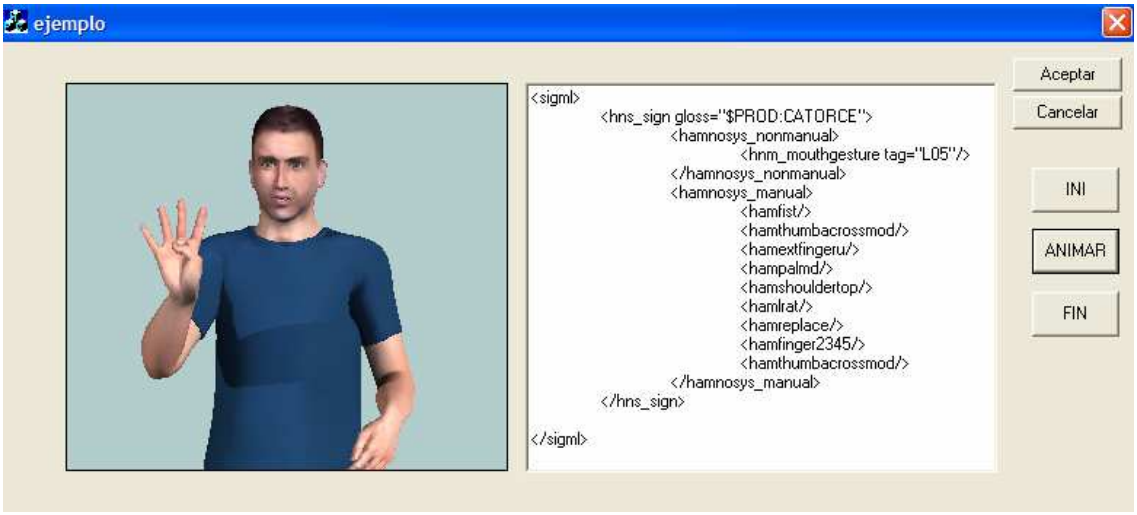


Figura 42: Programa *"ejemplo"* para representación de los signos

En la figura se muestra como el fichero .SiGML a representar se introduce en la parte derecha de la ventana, mientras que en la parte izquierda se visualiza VGuido. Para inicializar el programa y cargar VGuido, hay que pulsar el botón "INI", y para representar un signo, debe pulsarse el botón "ANIMAR". Si se pulsan los botones "Aceptar" o "Cancelar" se finaliza la representación y se cierra el programa.

A continuación se indican y describen brevemente los distintos ficheros fuente utilizados para implementar el programa.

- ***ejemplo.cpp***

Es el fichero fuente principal, que contiene la clase de la aplicación, *"CEjemploApp"*.

- ***ejemplo.rc***

En este fichero se encuentra una lista de todos los recursos que utiliza el programa. Incluye iconos, dibujos y cursores, que pueden ser editados directamente en Microsoft Visual C++.

- ***ejemplo.ico***

Este fichero contiene el icono que se utiliza como icono de la aplicación, y que está incluido en el fichero principal de recursos, *"ejemplo.rc"*.

- ***ejemplo.rc2***

Este fichero contiene los recursos que no pueden ser editados por Microsoft Visual C++. En este programa no se ha empleado ningún recurso modificado externamente, y por lo tanto, no se utiliza este fichero.

- ***hnstplayerctrl.cpp***

Este fichero contiene los métodos necesarios para visualizar y animar el agente virtual como control ActiveX. A continuación se describen brevemente estos métodos.

- ***GetFrameRate ().*** Un avatar genera una animación mediante una sucesión temporal de imágenes estáticas (*"frame"*), cada una de las cuales representa una postura del agente animado. Con el método *GetFrameRate* se obtiene la tasa de *frames* por segundo fijada para representar un signo.
- ***SetFrameRate (short nNewValue).*** Este método se utiliza para fijar una tasa de *frames* por segundo para la representación de los signos.
- ***PlayHNST (LPCTSTR bsHNST).*** Mediante este método, el agente animado realiza la animación del fichero .SiGML con la descripción HamNoSys correspondiente a un signo. Este fichero .SiGML está contenido en la cadena de caracteres que se le pasa al método como argumento.
- ***Replay ().*** Al llamar a este método el agente animado representa de nuevo la última animación realizada.
- ***ResetCamera ().*** Al ejecutar este método, la posición desde donde se observa el agente animado vuelve a la inicial.
- ***Initialise (long at).*** Con este método se carga el agente animado correspondiente. Se dispone de varios agentes además de VGuido, indicándose el avatar que debe cargarse a través del número que se le pasa al método como argumento. A VGuido le corresponde el número 6.
- ***PlaySiGML (LPCTSTR bsSiGMLIn).*** Mediante este método el agente animado representa el signo descrito en el fichero .SiGML contenido en la cadena de caracteres que se le pasa como argumento.
- ***SwitchAvatar (long at).*** Este método permite cambiar el agente animado visualizado.

- ***ejemploDlg.cpp***

La ventana de diálogo se genera a partir del fichero fuente *"ejemploDlg.cpp"*, que contiene la clase *"CejemploDlg"*. La plantilla de la ventana de diálogo se encuentra en el fichero de recursos *"ejemplo.rc"*, y puede ser editada con Microsoft Visual C++.

La clase *"CejemploDlg"* define el comportamiento de la ventana de diálogo, estableciendo las acciones a realizar cuando se pulsan los botones de "INI", "FIN", y "ANIMAR". Al pulsar el botón "INI" se carga el agente animado a través del método *"Initialise"*, definido en el fichero fuente *"hnstplayerctrl.cpp"*, pudiendo indicar en el código el agente virtual a visualizar.

Figura 43: Posibles agentes animados que se pueden visualizar

Esta posibilidad de elegir entre varios agentes es una muestra de la flexibilidad del sistema, aunque la representación de los signos es óptima con VGuido, que es el que mejores resultados consigue y el que más funcionalidades admite (expresión facial, movimientos de la boca, etc.).

Al pulsar el botón “ANIMAR” se llama al método *“PlaySiGML”*, definido en *“hnstplayerctrl.cpp”*, para representar el fichero SiGML contenido en la cadena de caracteres *“codigo_ejemplo”*. En esta variable se almacena el fichero .SiGML correspondiente al signo que interesa representar, a través del método *“carga_ejemplo”*, definido en *“ejemploDlg.cpp”*.

En el programa de representación de los signos implementado, se emplean ficheros .SiGML obtenidos directamente a partir de la transcripción HamNoSys del signo, pero cabe comentar que el software de VGuido incorpora un documento HTML que permite visualizar la representación de los signos descritos en un fichero SiGML Extendido. La representación de los signos mediante VGuido puede visualizarse a través del documento HTML “Camera-Control-Javascript-VGuido”, que se encuentra en Archivos de programa/eSIGN/SiGMLSigning, tras instalar el software del agente animado. En el archivo .SiGML “Camera-SiGML-Demo”, en Archivos de programa/eSIGN/ SiGMLSigning/Files, puede modificarse el signo a representar.

Para la generación de la descripción de un signo en SIGML se utilizaba el editor original del proyecto eSign. A lo largo de este proyecto se ha desarrollado una nueva versión de un editor que incorpora nuevas funcionalidades y que se describe en el siguiente apartado.

8. Entorno de diseño de los signos

El editor SEA-HamNoSys (SEA: Sistema de Escritura Alfabética; HamNoSys: Sistema de Notación de signo-escritura de la Universidad de Hamburgo) es una herramienta desarrollada en un principio para la creación de signos para la base de datos. Entendemos como creación de signos a la especificación en SiGML de la representación del correspondiente signo (lo que equivale prácticamente a codificar el signo en HamNoSys, ya que hay una relación unívoca). Debido a que en el ámbito español la especificación del SEA para la escritura de signos es la más extendida, ya existe un diccionario creado por la Fundación CNSE en el que aparece una gran cantidad de signos codificados en SEA. El agente virtual utilizado en el proyecto necesitaba la codificación en SiGML (equivalente a HamNoSys) para poder representar los signos, por lo que una herramienta capaz de realizar una conversión entre SEA y HamNoSys permitiría reaprovechar el esfuerzo realizado de la codificación en SEA del diccionario normativo. Ese es el motivo por el que se implementó esta herramienta, que aunque no es perfecta, en todos los casos sí responde adecuadamente, generando una plantilla en HamNoSys que luego se puede perfeccionar o mejorar.

En este capítulo vamos a describir las utilidades de las que dispone el editor, detallando en cada una de ellas cómo funcionan y cómo han sido programadas.

8.1. Utilidades disponibles en el Editor Sea-HamNoSys

El objetivo del editor es el de la creación de los archivos de texto con la codificación SiGML de cada signo. Un ejemplo de este tipo de archivo es el *ANTES.txt*, que contiene la codificación SiGML del signo etiquetado con la glosa ANTES. En la figura podemos ver el contenido de este archivo, siendo similar para los demás signos generados por el editor SEA-HamNoSys:

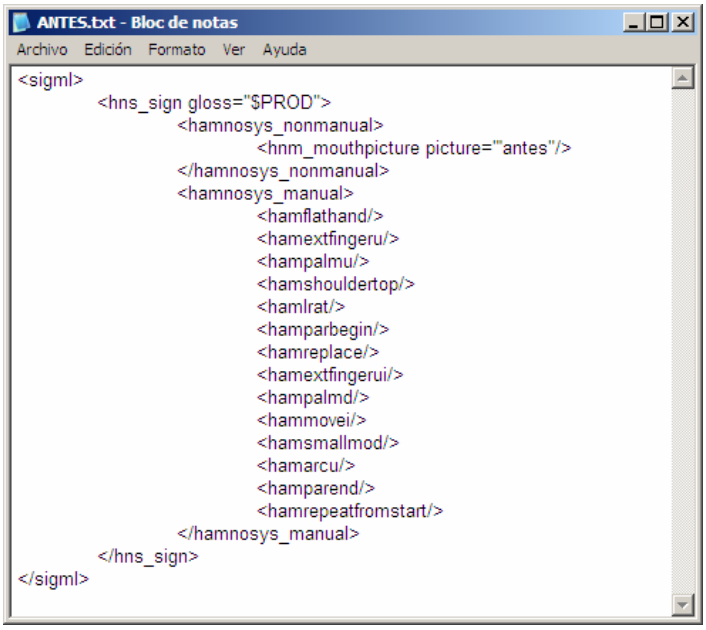


Figura 44: Contenido del archivo *ANTES.txt*

La creación de un archivo de este tipo es el objetivo del editor, aunque existen distintos modos de edición, para llegar finalmente al archivo SiGML. Estos son los caminos posibles para generar el archivo SiGML, cada uno de ellos comenzando con un modo de edición distinto:

- Edición en SEA→Conversión a HamNoSys→Conversión a SiGML→**Archivo**
- Edición en HamNoSys→Conversión a SiGML→**Archivo**
- Edición en SiGML directamente→**Archivo**

Los tres caminos están condicionados por el modo de edición elegido, por lo que dividiremos las utilidades según los modos de edición a los que pertenezcan. Además, existen dos utilidades comunes a los tres modos de edición, introducción de gestos y expresiones, y representación del signo por un agente animado. Por tanto, el conjunto de utilidades que vamos a describir en los apartados siguientes es:

- **Edición en SEA**
  - Inserción del signo especificado en SEA
  - Conversión SEA-HamNoSys
- **Edición en HamNoSys**
  - Inserción del signo especificado en HamNoSys
  - Conversión HamNoSys-SiGML
- **Edición en SiGML**
  - Apertura de archivo SiGML
  - Modificación del contenido SiGML
  - Creación del archivo SiGML
- **Introducción de gestos y expresiones**
- **Representación animada del signo**

8.2. Edición en SEA

Para la inserción de signos especificados en SEA utilizamos un cuadro de texto editable, que se encuentra en la parte superior derecha.

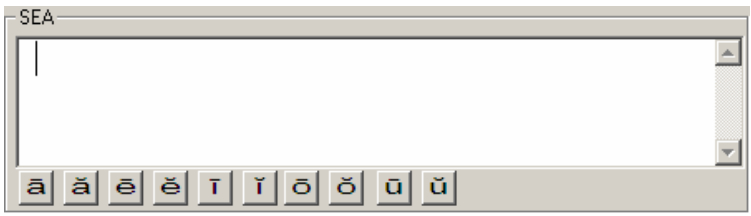


Figura 45: Cuadro de texto para la edición en SEA

Este cuadro de texto se maneja desde la aplicación principal (archivo *SeaHamSiGMLDlg.cpp*), mediante dos variables, una de tipo *CString* (**m_sea**) y otra de control (variable de tipo *CEdit* **m_sea2**).

Para codificar un signo en SEA se utilizan las letras del alfabeto, en conjunto con diversos acentos. Así, la mayoría de los signos pueden ser editados mediante un teclado corriente. Sin embargo, existen ciertas codificaciones que necesitan de acentos especiales. Debido a esto, se han añadido unos botones (se encuentran debajo del cuadro de texto, que podemos ver en la figura 2) específicos, para insertar los caracteres problemáticos, que no pueden ser editados mediante un teclado corriente.

Se han añadido diez botones especiales, que facilitan la inserción de los caracteres: ā, a, ē, ĕ, ī, ĭ, ō, ŏ, ū, ŭ. El tratamiento que se hace de cada uno de estos botones es similar, lo único que cambia es el carácter que edita en el cuadro de texto. Por ello, únicamente vamos a detallar el tratamiento de uno de los diez botones, en este caso el primero (situado más a la izquierda), que edita el carácter ā. Para manejar este botón se recoge el mensaje *ON_BN_CLICKED* de pulsación del botón, mediante la función *OnSEAchar1()* (para los demás botones únicamente cambia el número final, esto es, *OnSEAchar2*, *OnSEAchar3*…). Esta función realiza las siguientes operaciones:

- Sitúa el carácter ā en el lugar en el que se encuentra el cursor
- Coloca el foco en el cuadro de texto

Tabla 11: Esquema de *OnSEAchar1()*

Para situar el carácter se hace uso de la variable de control **m_sea2**, mediante la función *ReplaceSel*, que introduce el carácter indicado como parámetro en el lugar que apunta el cursor. La colocación del foco en el cuadro de texto se hace para que no se quede en el botón pulsado, pudiendo seguir escribiendo sin tener que pulsar sobre el cuadro de texto.

La fuente utilizada en el cuadro de texto es una fuente corriente modificada para que pueda sacar por pantalla estos caracteres especiales, utilizando caracteres ASCII. Esta fuente se denomina *sea*. Para activar esta fuente en el cuadro de texto, hacemos uso de la función *SetFont*, sobre el elemento del cuadro de texto (cuyo identificador es *IDC_SEAED*). Para crear la fuente hacemos uso de la función *CreateFontIndirect*, a la que tenemos que pasarle como parámetro una variable de tipo *LOGFONT*, donde hemos especificado el tipo de fuente (*sea*). La configuración de la fuente se realiza mediante la función *iniciaFuentes()*, que es llamada al inicio del diálogo, en *OnInitDialog()*.

Podemos editar signos con varias sílabas, separando cada uno de ellos mediante el carácter ‘-’.

8.2.1. Conversión SEA-HamNoSys

La conversión SEA-HamNoSys del texto que aparece en el cuadro de texto es llamada mediante la pulsación de la tecla ENTER, siempre que el foco siga sobre el propio cuadro de texto. Para realizar esto, debemos recoger los mensajes de pulsación de teclas del teclado, manejando la función

PreTranslateMessage(MSG pMsg)* del diálogo principal. Dentro de esta función, el tratamiento que se hace para la conversión SEA-HamNoSys es el siguiente:

- Se comprueba el mensaje *WM_KEYDOWN* de pulsación de tecla
  - Si coincide con la tecla ENTER
    - Si el foco se encuentra sobre el cuadro de texto de edición de SEA:
      - Lanzamos la conversión a HamNoSys
      - Situamos el foco sobre el cuadro de edición de HamNoSys

Tabla 12: Esquema de *PreTranslateMessage(…)*

Situamos el foco sobre el cuadro de texto de edición de HamNoSys porque se supone que, una vez realizada la conversón a HamNoSys, ya no interesa la edición en SEA, sino la posible modificación del resultado obtenido en la conversión. El lanzamiento de la conversión a HamNoSys se realiza mediante el llamamiento a la función *OnSeahns()*. Esta función copia en la variable **origen** el contenido del cuadro de texto de edición de HamNoSys, utilizando en ese caso la variable de tipo *CString* **m_sea**. El contenido de esta variable, que será la especificación en SEA que hemos editado, se le pasa a la función *seatoham(char*,char*)*, que es la encargada de realizar la conversión. Ésta será detallada en el capítulo correspondiente a la conversión SEA-HamNoSys.

Además de lanzar la conversión mediante la pulsación de la tecla ENTER, también podemos lanzarla pulsando el botón *Convertir SEA-HamNoSys*. Éste se encuentra debajo del cuadro de texto de edición en SEA.

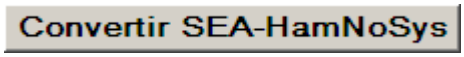


Figura 46: Botón *Convertir SEA-HamNoSys*

El mensaje de pulsación del botón, *ON_BN_CLICKED*, se recoge mediante la misma función *OnSeahns()* comentada anteriormente, realizando la misma función de llamada a *seatoham(char*,char*)*.

8.3. Edición en HamNoSys

Para editar en modo HamNoSys utilizamos un cuadro de texto editable similar al de edición en SEA que está justo debajo:

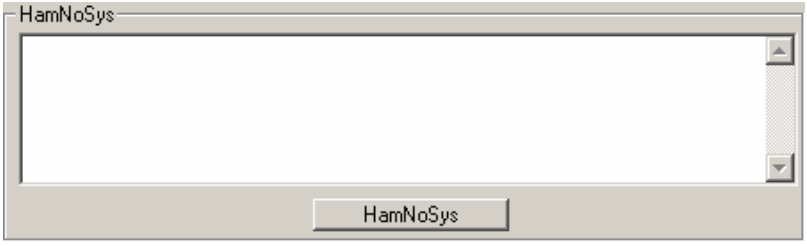


Figura 47: Cuadro de texto para la edición en HamNoSys

Al igual que el cuadro de texto de edición en SEA, éste necesita un tipo de fuente especial. En este caso el tipo de fuente es de tipo simbólico (debido al tipo de caracteres utilizados en la especificación de HamNoSys), por lo que necesitamos configurar el campo *lfCharSet* de la variable de tipo *LOGFONT* correspondiente, dándole el valor *SYMBOL_CHARSET* (este cambio es importante para la utilización de la fuente, ya que en el caso de no especificarlo utilizaría la fuente por defecto). Por lo demás, la configuración de la fuente es similar a la realizada para el caso del cuadro de edición de SEA, siempre dentro de la función *iniciaFuentes()*. En este caso la fuente utilizada se denomina *Hamnosys Plain*.

Debido a que los caracteres de HamNoSys son simbólicos y no pueden insertarse mediante un teclado corriente, necesitamos otro tipo de teclado especial. Este teclado especial lo lanzamos en una ventana secundaria, conteniendo botones para la inserción de cada uno de los caracteres. Para lanzar este teclado, necesitamos pulsar el botón *HamNoSys*, que se encuentra debajo del cuadro de texto de edición de HamNoSys, y que podemos ver en la figura siguiente.

Figura 48: Botón *HamNoSys*

La pulsación de este botón se trata mediante la función *OnHnskb()*, que recoge el mensaje *ON_BN_CLICKED* de pulsación del botón. Esta función realiza las siguientes operaciones:

• Si el diálogo de HamNoSys (ventana secundaria con el teclado) no se encuentra abierto:
o Se lanza en una nueva ventana el diálogo o Se activa el estado de teclado abierto

Tabla 13: Esquema de *OnHnskb()*

Para actualizar el estado del teclado, si está abierto o cerrado, utilizamos la variable booleana **tecladoAbierto**. Ésta se activa cuando lanzamos el diálogo con el teclado de HamNoSys. La comprobación de si el teclado está abierto es para que no se puedan lanzar innumerables diálogos. Cada vez que se cierra el diálogo, **tecladoAbierto** se pone a *false* nuevamente. Para indicar que el teclado se ha cerrado, se envía un mensaje a la aplicación principal, en este caso el mensaje *10002*. Éste se trata en el gestor de mensajes (mediante la función *DefWindowProc(UINT message, WPARAM wParam, LPARAM lParam)* del diálogo principal), poniendo **tecladoAbierto** a *false* cada que vez que se ha notificado.

Para lanzar el nuevo diálogo debemos crear un objeto de la clase correspondiente (que debe heredar de *CDialog*). Esta clase es *HamNoSys_KBDlg.cpp*. Para crear el diálogo se llama a la función *Create*, y para lanzarlo, a *ShowWindow*. Esta clase gestiona el nuevo diálogo, y es la encargada de tratar la selección de cualquier símbolo de HamNoSys, y de

notificarlo al diálogo padre. La estructura de recursos de este diálogo es la que especificamos en el siguiente esquema:

- Diálogo principal:
    - botón para eliminar caracteres insertados
    - botón para insertar un espacio
    - *tab control* con 6 pestañas, cada una contiene un nuevo diálogo:
        - Diálogo de *formas* con 32 botones
        - Diálogo de *orientaciones* con 34 botones
        - Diálogo de *localizaciones* con 52 botones
        - Diálogo de *movimientos* con 50 botones
        - Diálogo de *movimientos* con 41 botones
        - Diálogo de *mano pasiva* con 33 botones

En la figura podemos ver cómo es éste diálogo, en el que podemos apreciar la pestaña *formas*:

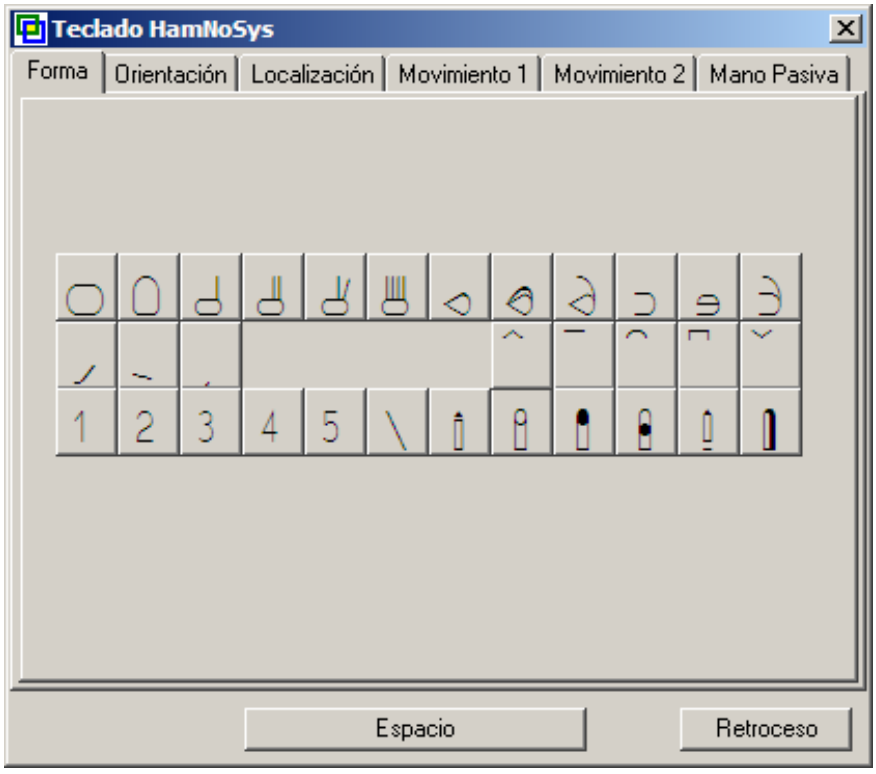


Figura 49: Diálogo con el teclado de HamNoSys

El botón de borrado tiene la etiqueta *Retroceso*, y se encarga de enviar un mensaje al diálogo padre (aplicación principal) indicando en cada pulsación que debe eliminarse el carácter anterior al puntero del ratón en el cuadro de edición de HamNoSys (equivale a la tecla *Supr* del teclado corriente). Al pulsar el botón, se notifica el mensaje *ON_BN_CLICKED*, que llama a la función

OnRetroceso(). Esta función únicamente envía un mensaje al diálogo padre, en este caso el mensaje **10001**, el cual se maneja en el gestor de mensajes de la aplicación principal.

Para insertar un espacio en el cuadro de texto de edición de HamNoSys se utiliza el botón *Espacio*. Cada vez que es pulsado, se introduce un espacio en el lugar donde se encuentra el cursor en el cuadro de edición. La pulsación se trata mediante la función *OnEspacio()*, que recoge el mensaje *ON_BN_CLICKED*. Esta función se encarga de enviar el mensaje **10000** a la aplicación principal, donde se maneja mediante el gestor de mensajes.

El recurso utilizado de tipo *tab control* tienes 6 pestañas y ocupa la mayor parte del diálogo de HamNoSys, a excepción de la zona inferior, donde se encuentran los dos botones antes mencionados. Este tipo de recurso se controla mediante una variable de tipo *CTabCtrl*, y permite realizar ciertas funciones, pero lo que no se puede realizar es controlar dentro de cada una de las pestañas otro diálogo distinto. Para conseguir esto hemos tenido que crear una nueva clase, *MyTabCtrl.cpp*, que hereda de *CTabCtrl* y que sobrescribe algunas de sus funciones.

Esta nueva clase realiza dos tareas fundamentalmente, iniciar los seis diálogos que se colocarán sobre las pestañas, y mostrar el correspondiente cuadro de diálogo cuando se haga un *click* sobre su pestaña.

La inicialización se realiza en parte en *MyTabCtrl.cpp* y también en la clase principal del diálogo de HamNoSys, *HamNoSys_KBDlg.cpp*, aunque todas las funciones que realizan esta tarea se encuentran implementadas en la primera. En el constructor de *MyTabCtrl.cpp* es donde se guardan los objetos diálogo de cada clase. El búfer donde se guardan los punteros de los seis objetos *CDialog* es **m_Dialog**. En este constructor es donde se crean las instancias de estos objetos, aunque la creación de los diálogos se realiza en la función *InitDialogs()*, también de *MyTabCtrl.cpp*. Esta función crea los seis diálogos mediante *Create*, y es llamada en la inicialización del diálogo de HamNoSys, en la función *OnInitDialog()* de la clase *HamNoSys_KBDlg.cpp*.

La otra tarea que se realiza en *MyTabCtrl,cpp* es la de mostrar el diálogo correspondiente según la pestaña seleccionada. Esta es la tarea más importante, ya que con ella se consigue poder manejar para cada pestaña un diálogo distinto. Para conseguirlo, se sobrescribe la función de *OnSelchange(NMHDR* pNMHDR, LRESULT* pResult)* de *CTabCtrl*, la clase que controlaría los recursos *tab control* por defecto. Esta función es la que trata el mensaje *TCN_SELCHANGE*, que notifica un cambio de selección en las pestañas de este recurso. Como la clase que estamos implementando hereda de *CTabCtrl*, para que este mensaje llegue a *MyTabCtrl.cpp*, a su vez hay que tratar el mensaje *ON_NOTIFY_REFLECT* dentro de esta última, que se encarga de recoger el anterior mensaje de cambio de selección en *CTabCtrl* y enviarlo a su vez al mapa de mensajes de *MyTabCtrl.cpp*. De esta manera conseguimos tratar el mensaje de cambio de selección en las pestañas del *tab control* en la nueva clase implementada. Dentro de *OnSelchange* se llama a la función *ActivateTabDialogs()*, que es donde se trata el diálogo correspondiente a la pestaña seleccionada.

Las operaciones que se realizan en esta función son principalmente éstas:

- Se obtiene la posición de la pestaña seleccionada.
- Se obtienen las dimensiones del *tab control.*
- Se colocan los seis diálogos sobre las dimensiones calculadas.
- Se muestra por pantalla el diálogo de la pestaña seleccionada, escondiendo los demás.

Tabla 14: Esquema de *ActivateTabDialogs()*

Para obtener las dimensiones del *tab control*, donde podremos colocar el diálogo, se hace uso de variables del tipo *CRect*, donde se guardan las dimensiones de un rectángulo. En este caso es necesario el uso de dos de estas variables, una para guardar las dimensiones absolutas del *tab control* (respecto del diálogo de HamNoSys), y la otra para las dimensiones relativas (tamaño del propio *tab control*). Las dimensiones absolutas las obtenemos mediante la función *GetWindowRect*, que las devuelve respecto de la pantalla, por lo que a su vez hay que relativizar al diálogo de HamNoSys. Estas dimensiones obtenidas nos dan la ventana completa del *tab control*, incluyendo las pestañas, por lo que es necesario obtener el rectángulo viable para colocar los diálogos, dejando fuera al bloque de pestañas. Debido a esto necesitamos una nueva variable *CRect* donde obtendremos las dimensiones del propio *tab control*, sin las pestañas, relativas al mismo.

Para obtener las dimensiones del rectángulo viable del *tab control*, necesitamos hacer uso de la función *AdjustRect*, a la que hay que pasarle como parámetro el rectángulo de la ventana completa del recurso, en dimensiones relativas al mismo. La propia función *AdjustRect* se encarga de tener en cuenta el bloque de pestañas. Obtenido el rectángulo viable respecto de la ventana completa del *tab control*, sólo queda relativizarlo al diálogo de HamNoSys, para lo que haremos uso de las dimensiones absolutas anteriormente calculadas.

Los diálogos se posicionan en el rectángulo viable obtenido, mediante la función *SetWindowPos*. Esta función permite pasarle como parámetro una etiqueta en la que podemos seleccionar si el diálogo debe esconderse o no. Para los diálogos no correspondientes a la pestaña seleccionada, se le pasa la etiqueta *SWP_HIDEWINDOW*. Al diálogo seleccionado se le pasa la etiqueta *SWP_SHOWWINDOW*. Para obtener la posición de la pestaña seleccionada utilizamos la función *GetCurSel()*, que devuelve la pestaña seleccionada actualmente.

El proceso de creación y lanzamiento del diálogo de HamNoSys lo podemos resumir en estos pasos:

- Lanzamiento del diálogo principal, al pulsar *OnTecladoHNS()*, manejado por la clase *HamNoSys_KBDlg.cpp*

- Éste contiene un *tab control*, con seis pestañas, del que sobrescribimos la función que trata el cambio de selección de pestañas, *OnSelchange*

  - Cada vez que seleccionamos una nueva pestaña, colocamos en el recuadro habilitado por el *tab control* los seis diálogos de HamNoSys (*IDD_DIALOG1*, *IDD_DIALOG3, IDD_DIALOG7, IDD_DIALOG4, IDD_DIALOG5, IDD_DIALOG6*), escondiendo los que corresponden a las pestañas no seleccionadas

Los seis diálogos de HamNoSys tienen una estructura similar, ya que únicamente se componen de botones, que se tratan igual en cada uno de ellos. La única diferencia radica en el número de botones que contiene cada uno de los diálogos, así como en el tipo de símbolos que se puedan seleccionar. Por tanto, únicamente vamos a detallar la estructura y el funcionamiento del primero de ellos, *IDD_DIALOG1*.

Este diálogo se maneja por una clase que hereda de *CDialog*, en este caso *MyDlg1.cpp*. Los dos puntos importantes que se tratan en ésta son: seleccionar la fuente adecuada para poder visualizar los símbolos, y tratar la pulsación de los botones, mediante la misma función para todos ellos.

La selección de la fuente se realiza al inicio del diálogo, dentro de la función *OnInitDialog()*, más concretamente en *iniciaFuentes()*. Para crear la fuente hacemos uso de la función *CreateFontIndirect*, que crea una variable de tipo *CFont* a partir de una *LOGFONT*. Para que ésta se configure correctamente debemos copiar en el atributo *lfFaceName* la cadena *"Hamnosys Plain"*, y en *lfCharSet* asignar la etiqueta *SYMBOL_CHARSET*, debido a que los caracteres de HamNoSys son de tipo simbólico.

Todos los botones se tratan de la misma manera, asociando la función *OnTecla()* a los mensajes *ON_BN_CLICKED* de todos los botones. El tratamiento que se realiza en esta función es el siguiente:

• Se obtiene el identificador del botón pulsado
• Se copia el texto que aparece sobre ese botón
• Se obtiene el carácter del símbolo seleccionado
• Se envía un mensaje con ese carácter al diálogo padre

Tabla 15: Esquema de *OnTecla()*

Para obtener el identificador del botón pulsado, se sigue el mismo procedimiento seguido en la aplicación principal en *OnRadioButton()*, esto es, se obtiene el foco mediante *GetFocus()*, ya que se habrá centrado sobre el último botón pulsado.

El texto que aparece en cada botón contiene normalmente un único carácter, el símbolo de HamNoSys correspondiente. Por tanto, para obtener el

carácter del símbolo debemos coger el primer carácter de la cadena. Sin embargo, en algunos botones se ha colocado un espacio antes del símbolo de HamNoSys, ya que algunos de estos símbolos únicamente tienen sentido si van detrás de otro símbolo, agrupándose únicamente en uno. Por eso, cuando sólo se muestra el símbolo sin ningún otro previo, éste no se puede visualizar, con lo que no sabríamos qué botón ha sido pulsado. Este el caso del símbolo que indica que el pulgar debe encontrarse perpendicular a la palma. Este símbolo debe ir añadido a uno de configuración de mano, como por ejemplo: . Aquí acompaña al símbolo de puño cerrado (). Sin embargo, si el símbolo del pulgar lo escribimos sólo, apenas se ve, ya que está pensado para ir siempre acompañando a otro. Si lo escribimos sin acompañar a ningún otro símbolo quedaría: .

El símbolo se alinea muy a la izquierda, por lo que en los botones no se ve, habiendo que escribir un espacio antes.

Estos símbolos están codificados en ASCII, pudiendo aparecer en el rango de 0 a 255. Sin embargo, al codificarse en *char* se utilizan además valores negativos, por lo que el rango real se encuentra entre -128 y 127. Al enviar el mensaje al diálogo padre, para que la asignación con la tabla de caracteres sea la correcta, los símbolos los convertimos al rango positivo.

El mensaje que se envía al diálogo padre (diálogo de la clase *HamNoSys_KBDlg.cpp*) es el **10000**. El símbolo pulsado de HamNoSys lo enviamos en la variable *wParam* (que le pasamos como parámetro en *SendMessage*). Como lo que queremos es que el mensaje llegue a la aplicación principal, para que se inserte el carácter indicado en el cuadro de texto de edición en HamNoSys, dentro de *HamNoSys_KBDlg.cpp* tenemos que recoger el mensaje **10000** y volverlo a enviar a su diálogo padre, que ya será el de la aplicación principal.

De esta manera podemos insertar cualquiera de los símbolos de HamNoSys disponibles, pudiendo en el cuadro de texto modificar el resultado de una conversión desde SEA, o simplemente crear uno nuevo, para posteriormente convertirlo en su equivalente SiGML.

8.3.1. Conversión HamNoSys-SiGML

La conversión HamNoSys-SiGML no se realiza de una manera explícita (no aparece un botón explícito para realizar dicha acción), sino que se realiza dentro de diversas operaciones, ya que es necesaria la conversión para poder realizarlas. Las operaciones en las que se necesita convertir el signo especificado en HamNoSys a su equivalente en SiGML son las siguientes:

- Reproducción animada del signo
- Modificación de la codificación en SiGML del signo
- Creación del archivo de texto con el contenido SiGML correspondiente

La reproducción animada del signo puede activarse de dos maneras distintas: mediante la pulsación de un botón, y mediante la pulsación de la tecla ENTER al encontrarnos en el cuadro de texto de edición en HamNoSys. En el apartado correspondiente a la reproducción animada del signo detallaremos estas dos operaciones.

Las otras dos operaciones en las que es necesaria la conversión HamNoSys-SiGML se refieren a utilidades del modo de edición en SiGML, que se detallarán en el apartado correspondiente a éstas.

Todas estas operaciones necesitan de la conversión HamNoSys-SiGML, y ésta se realiza con la llamada a la función *OnHns2sigml()*, hecho que se da en todas ellas. Para realizar la conversión, esta función realiza las siguientes tareas:

- Escribimos la cabecera SiGML en el búfer
- Para cada sílaba:
  - Para cada carácter HamNoSys de la sílaba:
    - Escribimos la etiqueta correspondiente al carácter en el búfer
- Escribimos el final SiGML en el búfer
- Para cada sílaba:
  - Llamamos a la inserción de los gestos especificados (estos se situarán en la parte no manual de la descripción en SiGML, justo encima de la parte manual)

Tabla 16: Esquema de *OnHns2sigml()*

```
<sigml>
    <hns_sign gloss="$PROD">
            <hamnosys_nonmanual>
                    <hnm_mouthpicture picture="apeL'iDo"/>
            </hamnosys_nonmanual>
            <hamnosys_manual>
                    <hamfinger2/>
                    <hamthumbopenmod/>
                    <hammiddlefinger/>
                    <hamfingerstraightmod/>
                    <hamextfingero/>
                    <hampalml/>
                    <hamchest/>
                    <hamlrat/>
                    <hamclose/>
                    <hamparbegin/>
                    <hamreplace/>
                    <hampinch12/>
                    <hammiddlefinger/>
```

```
                    <hamparend/>
                    <hamrepeatfromstartseveral/>
                </hamnosys_manual>
            </hns_sign>
        </sigml>
```

Tabla 17: Ejemplo de descripción SiGML. La parte no manual se encuentra coloreada en rojo

El búfer es una variable global donde en todo momento se guarda el contenido SiGML con el que se está trabajando, para poder modificarlo en cualquier momento, y finalmente escribirlo en el archivo final. El búfer se denomina **signo**.

A cada carácter en HamNoSys le corresponde una etiqueta SiGML. Todas estas etiquetas se guardan en una tabla, que es una variable de tipo array de *char*, y que se denomina **tabla**. Ésta se carga al inicio de la aplicación, en *OnInitDialog()*, mediante la llamada a la función *cargaTabla()*. La tabla contiene 256 elementos, cada uno correspondiente a un carácter ASCII. Los primeros 34 elementos (caracteres ASCII del 0 al 33) no equivalen a ninguna etiqueta, por lo que se igualan a cero. En los demás se carga la etiqueta correspondiente.

Diferenciamos las sílabas mediante un espacio en la edición en HamNoSys. Al convertirlas a SiGML, cada sílaba se trata como una descripción de signo diferente, por lo que hay que ir concatenando cada signo (sílaba) uno detrás de otro en el mismo código SiGML. Sin embargo, en vez de escribir cada codificación en SiGML por separado, sólo concatenamos el contenido entre cabeceras de cada signo. Así, únicamente ponemos una cabecera SiGML al principio, y la etiqueta final SiMGL al final. De esta manera, la apariencia de la codificación SiGML es la de un único signo, aunque dentro hemos concatenado todos los correspondientes a las sílabas editadas. Codificando así las sílabas, conseguimos que el agente virtual concatene los signos sin resetear el movimiento cada vez, esto es, sin bajar las manos a la posición inicial, con lo que la concatenación de los signos es más natural.

Dentro de la codificación SiGML también se especifican los gestos y expresiones del cuerpo, que escribimos en **signo** al final de la función *OnHns2sigml()*. Para escribir los gestos y expresiones correspondientes de cada una de las sílabas, debemos llamar a la función *ponGesto*. El funcionamiento de esta función se detalla en el apartado correspondiente a gestos y expresiones del cuerpo.

8.4. Edición en SiGML

La edición en SiGML se compone de tres utilidades principales:

- Apertura de un archivo SiGML
- Modificación del contenido SiGML
- Creación del archivo SiGML

El manejo de todas ellas se basa en el búfer **signo** comentado anteriormente, el cual guarda el contenido en SiGML actual con el que se está trabajando, ya sea por la apertura de un archivo o modificación del mismo.

8.4.1. Apertura de un archivo SiGML

Podemos cambiar el contenido de **signo** mediante la apertura de un archivo SiGML, volcándose su contenido en el búfer. Para realizar esta operación hacemos uso de un botón del interfaz principal, situado debajo del cuadro de texto de edición en HamNoSys, en el cuadro correspondiente a la parte *SiGML*. Éste es el botón *Abrir…*.

Figura 50: Botón *Abrir…*

La pulsación de este botón se maneja mediante la función *OnAbrir()*, que recoge el mensaje de pulsación de botón *ON_BN_CLICKED*. Esta función realiza las siguientes operaciones:

- Creamos un diálogo gestor de archivos, filtrando los archivos de texto
- Lanzamos el nuevo diálogo en una ventana secundaria:
  - Si se ha cerrado pulsando *Aceptar*:
    - Recogemos la ruta del archivo especificado
  - Si se ha cerrado pulsando *Cancelar*:
    - Nos salimos de la función
- Si el archivo de texto es válido:
  - Volcamos su contenido en el búfer **signo**
- Convertimos el contenido SiGML a HamNoSys

Tabla 18: Esquema de *OnAbrir()*

El diálogo gestor de archivos es de tipo *CFileDialog*. Para filtrar los archivos que muestra, dejando únicamente los de texto, debemos configurar el atributo *m_ofn.lpstrFilter* de la variable de tipo *CFileDialog*.

Si el diálogo se cierra mediante la pulsación de *Aceptar*, recogemos la ruta especificada en el diálogo, mediante la función *GetPathName()*, que llamamos desde la variable manejadora del nuevo diálogo, de tipo *CFileDialog*.

Para volcar el contenido del archivo especificado en el nuevo diálogo, llamamos a la función *carga_ejemplo(char * prueba)*. Esta función se encarga de abrir el archivo cuya ruta es *prueba*, y volcar su contenido en el búfer **signo**. Una vez modificado correctamente el contenido SiGML de **signo**, llamamos a la conversión equivalente a HamNoSys, para que aparezca en el cuadro de texto de edición de HamNoSys, y así poder modificarlo si se desea. La función

que realiza la conversión es *OnSigml2hns()*, y se estructura de la siguiente manera:

- Copiamos el contenido de **signo** en otro búfer, para ir mirando sobre este último (de esta manera no modificamos el original **signo**, ya que hacemos uso de la función *strtok*, que modifica el búfer sobre el que trabaja)
- Mientras la etiqueta sea distinta de la indicadora de parte manual:
  - Se mira la siguiente línea
- Mientras nos encontremos en la parte manual del código SiGML:
  - Se quitan tabulaciones hasta encontrar la etiqueta
  - Se compara la etiqueta con las etiquetas de la tabla de HamNoSys
  - Se copia el carácter correspondiente a la etiqueta
  - Se mira la siguiente línea
- Si la secuencia es válida (todas las etiquetas miradas se encontraban en la tabla) se copia en el cuadro de texto de HamNoSys para que aparezca por pantalla.

Tabla 19: Esquema de *OnSigml2hns()*

La parte manual, etiquetas correspondientes a la codificación en HamNoSys, se encuentra entre las etiquetas **<hamnosys_manual>** y **</hamnosys_manual>**. Cada línea contiene una etiqueta SiGML, que a su vez corresponde con un carácter de HamNoSys. Para obtener el carácter correspondiente, debemos mirar en la tabla de etiquetas SiGML, que también utilizamos en la conversión de HamNoSys a SiGML, y cuyo nombre de variable es **tabla**.

Una vez obtenidos todos los caracteres equivalentes en HamNoSys a partir de las etiquetas de **tabla**, lo copiamos en el cuadro de texto de edición en HamNoSys. Para ello copiamos el contenido de HamNoSys en la variable de tipo *CString* **m_ham**, asociada al cuadro de texto. Para que aparezca por pantalla debemos llamar a la función *UpdateData(FALSE)*.

8.4.2. Modificación del contenido SiGML

En cualquier momento podemos modificar el contenido actual SiGML, que corresponde al contenido del búfer **signo**. Para poder editar este contenido, hacemos uso de un diálogo auxiliar de edición de texto, que cargará el contenido de **signo** para poder modificarlo. Para cargar este diálogo de edición debemos pulsar el botón *Editar…*, que se encuentra en el cuadro correspondiente de *SiGML*.

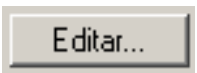


Figura 51: Botón *Editar…*

La pulsación de este botón se trata mediante la función *OnEditar()*, que recoge el mensaje de pulsación de botón *ON_BN_CLICKED*. Esta función realiza las siguientes operaciones:

- Actualizamos **signo** con el contenido del cuadro de texto de HamNoSys
- Copiamos **signo** en otro búfer, con el formato adecuado para el diálogo de edición
- Lanzamos el diálogo de edición de texto:
  - Si se ha cerrado mediante *Aceptar*:
    - Volcamos el texto del diálogo de edición en **signo**
    - Convertimos a HamNoSys para que aparezca por pantalla
  - Si se ha cerrado mediante *Cancelar*:
    - Volvemos a editar el cuadro de HamNoSys con el contenido inicial

Tabla 20: Esquema de *OnEditar()*

Lo primero que debemos hacer es actualizar **signo**, ya que hemos podido modificar la especificación de HamNoSys, por lo que el contenido SiGML actual ha cambiado también. La actualización se realiza de la misma manera que en la apertura de ficheros SiGML, mediante la llamada a la función *OnHns2sigml()*.

El diálogo de edición de texto se maneja mediante una variable de la clase *CEditorDlg.cpp*. El único recurso que contiene este diálogo (además de los botones por defecto *Aceptar* y *Cancelar*) es un cuadro de texto editable sobre el que aparecerá el contenido SiGML.

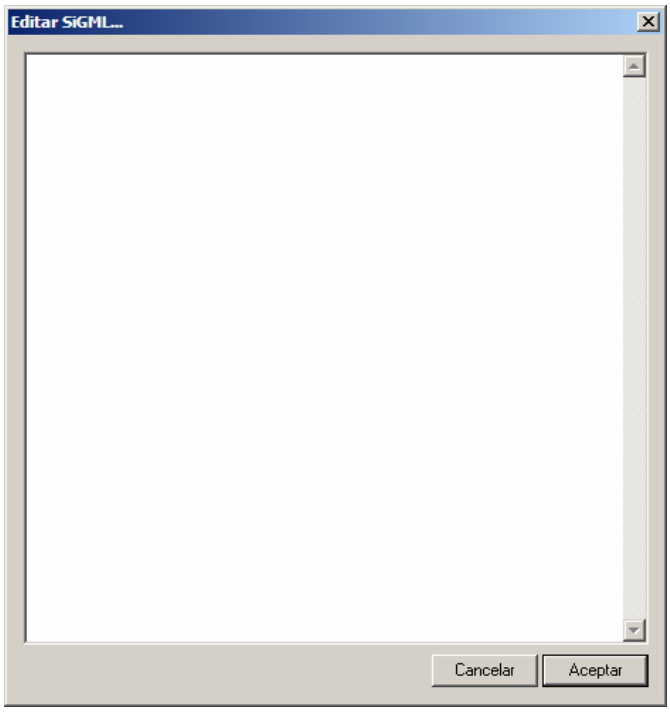


Figura 52: Diálogo de edición de texto

El cuadro de texto editable tiene asociada una variable de tipo *CString*, denominada **m_texto**, además de una variable de control, **m_texto2**.

Siguiendo con la función *OnEditar()*, debemos copiar **signo** en el cuadro de texto del diálogo de edición. Para ello debemos copiarlo en **m_texto**, aunque con el formato adecuado. Para que aparezca cada línea de **signo** en una línea diferente en el cuadro de texto, debemos añadir el carácter '\r' al final de cada línea de **signo**. Es necesario añadir este carácter, ya que en la lectura de cualquier fichero en Windows éste no aparece, pero sin embargo es necesario que aparezca en los cuadros de texto, ya que si no apareciera no habría cambio de línea. Una vez realizado esto, se actualiza **m_texto**.

El diálogo lo lanzamos mediante *DoModal()*, cuyo valor de retorno debemos recoger al cerrarse el diálogo, para comprobar si ha sido cerrado mediante la pulsación del botón *Aceptar*. Si es así, quiere decir que las modificaciones realizadas en el diálogo de edición deben ser guardadas, por lo que el contenido del cuadro de texto (**m_texto**) debemos copiarlo en **signo** para actualizarlo nuevamente, deshaciendo el formato anterior (ahora hay que quitar los caracteres '\r'). Una vez actualizado **signo**, convertimos su contenido SiGML a HamNoSys mediante *OnSigml2hns()*, que sacará por pantalla (cuadro de texto de edición en HamNoSys) el nuevo contenido modificado en el cuadro de edición de texto.

8.4.3. Creación del archivo SiGML

La única utilidad del modo de edición en SiGML que queda por describir es la de creación del archivo SiGML. Mediante ésta, volcamos el contenido SiGML actual (contenido equivalente HamNoSys-SiGML con el que estamos trabajando) en un archivo de tipo texto (*.txt*).

Para realizar la creación debemos pulsar el botón *Guardar…*, que se encuentra en el cuadro correspondiente SiGML.

Figura 53: Botón *Guardar…*

La pulsación de este botón se maneja mediante la función *OnGuardar()*, que recoge el mensaje de pulsación de botón *ON_BN_CLICKED*. La estructura de tareas de esta función es la siguiente:

- Creamos un diálogo gestor de archivos, filtrando los archivos de texto
- Actualizamos el búfer **signo**
- Lanzamos el diálogo gestor de archivos:
  - Si se ha cerrado mediante el botón *Aceptar*:
    - Recogemos la ruta especificada
  - Si se ha cerrado mediante el botón *Cancelar*:

▪ Nos salimos de la función

• Abrimos el archivo de la ruta especificada y volcamos el contenido de **signo**

Tabla 21: Esquema de *OnGuardar()*

La creación del diálogo gestor de archivos y la filtración de los archivos con extensión de texto se realiza de la misma manera que en la función *OnAbrir()*, manejando una variable de tipo *CFileDialog*.

Debemos actualizar **signo**, por si el contenido del cuadro de texto de edición en HamNoSys ha sido modificado, lo que realizamos llamando a la función *OnHns2sigml()*.

La ruta especificada en el diálogo gestor de archivos la recogemos mediante la llamada a la función *GetPathName()*, que posteriormente copiaremos en la cadena de texto **archivo_sigml**.

Para crear el archivo de texto, llamamos a la función *guarda_ejemplo(char * prueba)*, que le pasamos en **prueba** el contenido de **archivo_sigml**, que es la ruta especificada para la creación del archivo de texto. La función *guarda_ejemplo* se encargar de crear el archivo de texto. Para ello abre en modo de escritura el archivo especificado en la ruta **archivo_sigml** y escribe línea a línea el contenido de **signo**.

De esta manera creamos los archivos SiGML, objetivo final del editor SEA-HamNoSys, para ir generando la base de datos de signos necesaria en el proyecto.

8.5. Introducción de gestos y expresiones

Además de codificar en SiGML el equivalente a los caracteres de HamNoSys especificados para cada signo, la propia especificación de SiGML (creada expresamente para la representación por parte del agente virtual VGuido) permite la introducción de ciertas expresiones en la cara y en el cuerpo, que se representan por el agente virtual a la vez que el propio signo.

Existen seis tipos de expresiones que podemos insertar en la codificación SiGML:

- Expresiones en la boca: dientes, mandíbula, labios, pómulos, lengua y pronunciación con los labios
- Expresiones con los hombros: subirlos, echarlos hacia delante o subirlos y bajarlos (concatenando los dos movimientos)
- Expresiones con la mirada: hacia arriba, hacia abajo, hacia la izquierda y hacia la derecha
- Expresiones con las cejas
- Expresiones con los párpados
- Expresiones con la nariz

Para insertar estas expresiones, hacemos uso de un diálogo en una ventana secundaria, donde se encuentran las utilidades necesarias para poder insertarlas. Cada vez que insertemos alguna expresión, éstas se quedan guardadas, apareciendo siempre en el código SiGML hasta que sean borradas. Además, para comprobar la inserción de expresiones, dentro del interfaz principal existe un campo sobre el estado de las expresiones, que nos mostrará la última inserción realiza. Por tanto, existen tres elementos principales en la inserción de expresiones en la cara y en el cuerpo, relacionadas entre sí:

- Las expresiones insertadas quedan guardadas, apareciendo sobre el contenido actual con el que trabajamos, contenido de **signo**.

```
<sigml>
    <hns_sign gloss="$PROD">
            <hamnosys_nonmanual>
                    <hnm_mouthpicture picture="'a"/>
            </hamnosys_nonmanual>
            <hamnosys_manual>
                    <hamfist/>
                    <hamthumbacrossmod/>
                    <hamextfingeru/>
                    <hampalmd/>
                    <hamshoulders/>
                    <hamlrat/>
            </hamnosys_manual>
    </hns_sign>
</sigml>
```

Tabla 22: Contenido SiGML del signo A, en rojo las expresiones insertadas

- Para realizar alguna inserción, utilizamos una nueva ventana.

Figura 54: Diálogo para la inserción de expresiones de cara y cuerpo

- Cuadro de estado de expresiones, en el interfaz principal, donde se muestra la última inserción realizada (en cada inserción debemos especificar la sílaba en la que insertar las expresiones, 1 sílaba→1 inserción, cada vez). En este cuadro se encuentra también el botón que lanza el diálogo de inserción de expresiones, *Añadir*.

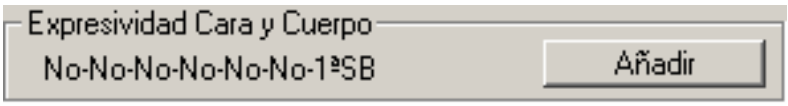


Figura 55: Cuadro de estado de expresividad cara y cuerpo

El diálogo de inserción de expresiones está implementado de la misma manera que el diálogo de HamNoSys (que describimos en el apartado de edición de HamNoSys), esto es, el diálogo contiene un recurso de tipo *tab control*, cuya clase manejadora (*MyTabCtrl2.cpp*) ha sido sobrescrita para poder insertar diálogos dentro de cada pestaña del *tab control* (en este caso 3 diálogos en vez de 6, por eso hemos tenido que implementar una nueva clase manejadora del *tab control*). Por tanto, existen tres niveles de ventanas:

Figura 56: Esquema de interacción de ventanas y mensajes

Como podemos ver en la figura, los mensajes siguen el siguiente camino: comienzan en los diálogos (manejados por las clases *Gesto0.cpp*, *Gesto1.cpp* y *Gesto2.cpp*) incluidos en las pestañas del *tab control*, al insertar o modificar las expresiones. De ahí llegan al diálogo de expresividad cara y cuerpo (manejado por la clase *HNS_gestosDlg.cpp*), donde son tratados y enviados a su vez al diálogo del interfaz principal (manejado por la clase principal, *SeaHamSiGMLDlg.cpp*). Todos los mensajes siguen este camino, excepto el que hace referencia a la pronunciación con labios, que es algo especial y describiremos en detalle.

Debido a que la implementación es similar a la realizada en el teclado de HamNoSys, en esta descripción nos vamos a centrar en el protocolo de mensajes de Windows que es un aspecto de implementación muy importante. Como el camino de éstos comienza en los diálogos contenidos en las pestañas, comenzaremos por uno de éstos, poniendo como ejemplo uno de sus recursos, ya que todos los demás (tanto los de su mismo diálogo como los pertenecientes a los otros dos diálogos contenidos en las pestañas) presentan un funcionamiento equivalente.

Los recursos de los diálogos presentan un funcionamiento equivalente, dividiéndolos en seis grupos, uno por cada tipo de expresión disponible. De

esta forma sólo se puede especificar un aspecto dentro de cada grupo: por ejemplo, si especificamos una pronunciación de labios, no podemos especificar al mismo tiempo una configuración de los dientes (ambos se encuentran en el grupo 1). Así, la relación de los 6 grupos de recursos es:

- Diálogo de *Boca*:

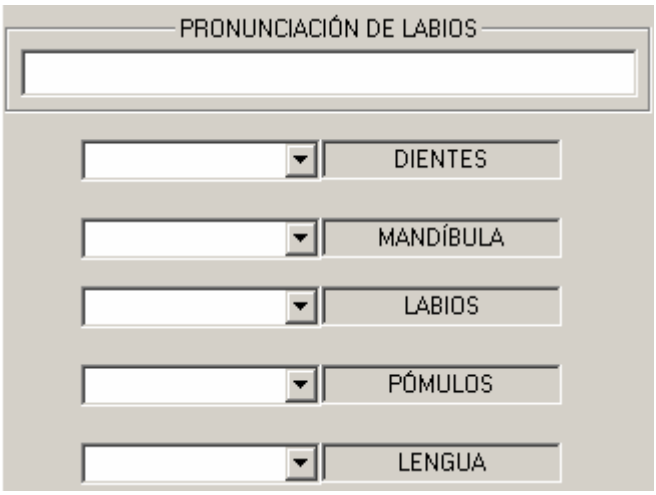


Figura 57: Recursos del diálogo *Boca*

  - **GRUPO 1** → Cuadro de texto *PRONUNCIACIÓN DE LABIOS*, lista desplegable *DIENTES*, lista desplegable *MANDÍBULA*, lista desplegable *LABIOS*, lista desplegable *PÓMULOS*, lista desplegable *LENGUA*

- Diálogo de *Cuerpo*:

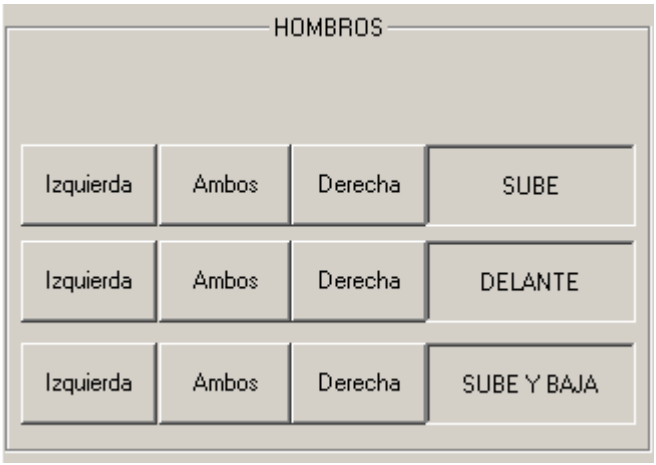


Figura 58: Recursos del diálogo *Cuerpo*

  - **GRUPO 2** → Botón *SUBE Izquierda*, Botón *SUBE Derecha*, Botón *SUBE Ambos*, Botón *DELANTE Izquierda*, Botón *DELANTE Derecha*, Botón *DELANTE Ambos*, Botón *SUBE Y BAJA Izquierda*, Botón *SUBE Y BAJA Derecha*, Botón *SUBE Y BAJA Ambos*

- Diálogo de *Cara*:

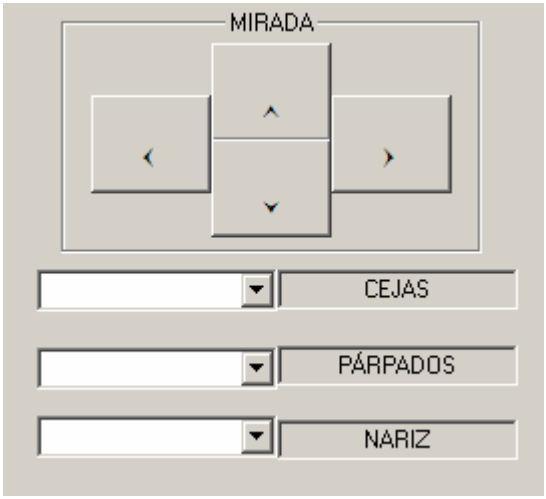


Figura 59: Recursos del diálogo *Cara*

  - **GRUPO 3** → Botón *MIRADA ARRIBA*, Botón *MIRADA ABAJO*, Botón *MIRADA IZQUIERDA*, Botón *MIRADA DERECHA*
  - **GRUPO 4** → Lista desplegable *CEJAS*
  - **GRUPO 5** → Lista desplegable *PÁRPADOS*
  - **GRUPO 6** → Lista desplegable *NARIZ*

Los recursos de los tres diálogos están divididos de esta manera porque al codificarlos en SiGML, únicamente podemos insertar una expresión por cada grupo, pudiendo por tanto insertar en cada signo hasta seis expresiones (una por grupo).

Para describir el protocolo de mensajes de Windows desde arriba (diálogos de pestañas), vamos a detallar un ejemplo, en este caso el recurso lista desplegable *DIENTES* del diálogo *Boca* (el recurso del cuadro de texto *PRONUNCIACIÓN DE LABIOS* tiene un tratamiento especial, con lo que se describe por separado).

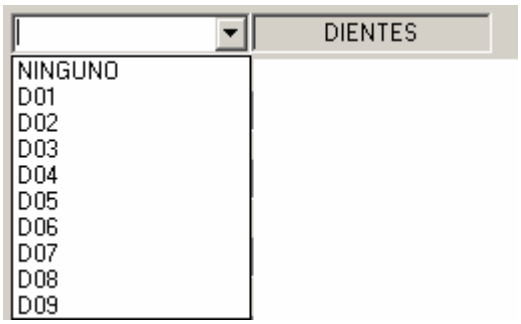


Figura 60: Lista desplegable *DIENTES*

La lista desplegable *DIENTES*, al igual que las demás, se inicializa al comenzar el diálogo (en este caso *Gesto0.cpp*), dentro de la función

OnInitDialog(). La función que realiza la inicialización es *iniciaListas()*. Esta función se encarga de añadir elementos a la lista correspondiente. Los datos de las listas se encuentran en varias tablas. En este caso, necesitamos dos tablas: **boca**, donde encontramos las iniciales de los elementos (en este caso D de dientes), y **bocaNum**, donde encontramos el número de elementos a insertar en cada lista (en este caso 9). Todas las tablas se encuentran en *Gesto0.cpp*, incluyendo las correspondientes a los recursos de los otros dos diálogos. Esta lista desplegable, al igual que las otras cuatro del diálogo *Gesto0.cpp*, se tratan mediante la recogida de dos mensajes:

- ON_CBN_DROPDOWN: se envía cada vez que se despliega una lista. Este mensaje en todas las listas de *Gesto0.cpp* se recoge mediante la función *OnDropdown()*.

- ON_CBN_SELCHANGE: se envía cada vez que se modifica la selección en una de las listas. En el caso de *DIENTES*, la función que recoge el mensaje es *OnComboD()*. En las otras listas de *Gesto0.cpp* lo único que cambia es la letra final en la función (D en este caso, por dientes).

La función *OnDropdown()* se encarga de borrar la selección en todas las listas, ya que en este caso, sólo podemos insertar una expresión en todo el diálogo *Boca*. Por ello, al desplegar una lista, que se supone que es para insertar una expresión de la misma, las otras selecciones no tienen sentido, por lo que se realiza un borrado de todas. Esto se hace mediante la función *SetCurSel* sobre el manejador de cada lista, configurándolo en la posición -1, lo que hace que no aparezca ninguno de los elementos de la lista.

La función *OnComboD()* se encarga de tratar el cambio de selección en la lista desplegable *DIENTES*. Esta función se encarga de recoger el índice del elemento seleccionado, mediante el manejador de la lista, utilizando la función *GetCurSel()*. Este índice es enviado en un mensaje a su ventana padre, en este caso la del diálogo de expresiones (*HNS_gestosDlg.cpp*), utilizando la función *SendMessage*, mediante el mensaje **10010**, enviando en el parámetro *lParam* el índice obtenido, y en el parámetro *wParam* el tipo de expresión (para diferenciar entre los 6 tipos de expresiones existentes, se envía en el 1º byte, en este caso un 0, expresiones del 1º grupo) y el tipo de recurso (se envía en el 3º byte, en este caso un 0, ya que es la primera lista desplegable).

En este nivel (**primer nivel**), diálogos *Gesto0.cpp*, *Gesto1.cpp* y *Gesto2.cpp*, se tratan 3 tipos de mensajes, 2 salientes (hacia el padre) y 1 entrante (procedente del padre):

- Mensajes salientes:

    - **10010** → Se ha seleccionado alguna expresión. Para diferenciar cada una de las expresiones en los 3 diálogos, se sigue la codificación de la tabla siguiente:

TRATAMIENTO DEL MENSAJE **10010**				
Diálogo	Grupo de expresión	Tipo de recurso	*wParam*	*lParam*
Gesto0.cpp	1º Grupo *wParam*=0xXX**00**	*DIENTES*	0x**00**00	Índice del elemento
		MANDÍBULA	0x**01**00	Índice del elemento
		LABIOS	0x**02**00	Índice del elemento
		PÓMULOS	0x**03**00	Índice del elemento
		LENGUA	0x**04**00	Índice del elemento
		PRONUNCIACIÓN DE LABIOS	0x**06**00	1 si se ha editado texto
Gesto1.cpp	2º Grupo *wParam*=0xXX**01**	9 botones para expresiones en los hombros	0x0001	Índice del botón
Gesto2.cpp	3º Grupo *wParam*=0xXX**02**	4 botones para expresiones de la mirada	0x0002	Índice del botón
	4º Grupo *wParam*=0xXX**03**	*CEJAS*	0x0003	Índice del elemento
	5º Grupo *wParam*=0xXX**04**	*PÁRPADOS*	0x0004	Índice del elemento
	6º Grupo *wParam*=0xXX**05**	*NARIZ*	0x0005	Índice del elemento

Tabla 23: Tratamiento del mensaje 10010

- **10007** → Se ha pulsado el botón *Video de muestra*, por lo que hay que lanzar el vídeo en un programa externo. Para ver estas expresiones, además se puede pulsar el botón *Ayuda*, que lanza un documento en formato *Word*, donde se encuentran imágenes de cada expresión.

- Mensajes entrantes:

  - **10009** → Se ha aceptado la selección, la inserción de expresiones es válida, por lo que hay que comprobar si en *PRONUNCIACIÓN DE LABIOS* se ha insertado alguna expresión, quedando como única posible en el diálogo *Boca*. Se trata mediante la función *OnAplicar()*. Ésta comprueba si en el cuadro de texto se ha editado algo. En caso afirmativo se envía al padre el mensaje

correspondiente de *PRONUNCIACIÓN DE LABIOS* **10010**. El texto editado se guarda en la variable **labios**, que será leída en su momento por las clases de los diálogos padres.

- **10005** → Se deben borrar todas las selecciones, por lo que hay que configurar todas las listas en la posición -1.

En el **segundo nivel**, esto es, en el diálogo de expresiones (*HNS_gestosDlg.cpp*), se tratan los siguientes mensajes:

- Mensajes salientes (hacia el padre):
  - **10011** → Ha habido un cambio válido en las expresiones o en la sílaba a insertar, el cuadro de estado del interfaz principal debe ser modificado.
  - **10003** → El diálogo ha sido cerrado.
  - **10007** → Se envía la última selección del primer tipo de expresión (las del diálogo *Boca*), para que se lance el programa de vídeo con esa expresión. Las selecciones realizadas temporalmente en los diálogos *Gesto0.cpp*, *Gesto1.cpp* y *Gesto2.cpp* se guardan en la variable **gestosAct**.
  - **10010** → La selección temporal se ha validado, por lo que se comunica a la ventana padre (interfaz principal) para que se queden guardadas. Hay dos tipos de validaciones: **validado de expresiones temporales realizadas** y **borrado de todas las expresiones guardadas**.
  - **10003** → El diálogo de expresiones se ha cerrado.
- Mensajes salientes (hacia a alguno de los hijos, diálogos *Gesto0.cpp*, *Gesto1.cpp* o *Gesto2.cpp*):
  - **10005** → Se indica que se han borrado todas las selecciones, se envía a los tres diálogos para que los recursos aparezcan vacíos.
  - **10009** → El diálogo se ha cerrado validando la selección realizada, esto es, pulsando el botón *Aceptar*. Se envía al diálogo *Gesto0.cpp* para que compruebe si el texto *PRONUNCIACIÓN DE LABIOS* ha sido editado, ya que predominará sobre la posible selección realizada en los demás recursos del diálogo *Boca*.
- Mensajes entrantes:

- **10010** → Se ha realizado alguna selección en alguno de los tres diálogos, por lo que hay que guardarlas temporalmente, en **gestosAct**.

- **10007** → Se ha pulsado el botón *Video de muestra* en *Gesto0.cpp*, por lo que se indica al interfaz principal, enviando la selección temporal del diálogo *Boca*, mediante **gestosAct**.

Cada vez que se ha realizado una selección en alguno de los 3 diálogos incluidos en las pestañas, ésta se notifica al diálogo de expresiones mediante el mensaje **10010**, como hemos comentado. Al recibirlas en *HNS_gestosDlg.cpp*, se van guardando como selecciones temporales en **gestosAct**. Esta variable es un array de 6 (grupos) x 2 (bytes). En el primer índice guardamos el tipo de expresión seleccionada (de entre los 6 grupos posibles), por lo que debemos obtener el módulo 256 del número guardado en *wParam*, ya que el tipo de expresión se guardaba en los 2 primeros bytes. El segundo índice sirve para guardar *wParam* y *lParam*, 0 y 1 respectivamente.

Cuando se pide un video de muestra desde el diálogo *Boca*, debe mostrarse el vídeo de la última expresión seleccionada en este diálogo. Esta selección es temporal, ya que el diálogo de expresiones no se ha cerrado y por lo tanto no se han validado las expresiones insertadas. La última selección del diálogo *Boca*, por tanto, debido a que se refiere al primer grupo de expresiones, se encuentra en **gestosAct[0][0]** y **gestosAct[0][1]**, que es lo que enviamos junto con el mensaje **10007**.

Dentro de este segundo nivel, tratado desde el diálogo de expresiones (*HNS_gestosDlg.cpp*), quedan por tratar sus propios recursos, que a su vez interaccionan en el envío general de mensajes. Estos recursos son:

- Botones de sílaba donde insertar las expresiones. Se tratan recogiendo el mensaje de pulsación de botón, mediante la función *OnRadio()*.

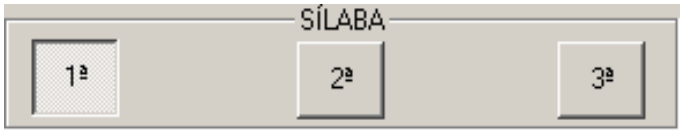


Figura 61: Botones de selección de sílaba

- Botón de borrado de selecciones, tanto de las guardadas como de las temporales. El botón se trata mediante la función *OnBorrar()*.

Figura 62: Botón *Borrar todo*

- Botón de validación de selecciones, *Aceptar*. Se trata mediante *OnCerrarg()*.

Figura 63: Botón *Aceptar*

- Botón de cerrado del diálogo de expresiones, sin validar las selecciones realizadas, *Cancelar*. Se trata mediante la función *OnCancelar()*.

Figura 64: Botón *Cancelar*

La función *OnRadio()* se encarga de recoger el tipo de sílaba seleccionada, obteniendo el texto de la etiqueta de cada botón *GetWindowText*, ya que el primer carácter es el número deseado. El número de sílaba seleccionada se guarda en la variable **silb**. Posteriormente se envía el mensaje **10011** al interfaz principal, que indica un cambio en el cuadro de estado. Cada vez que se selecciona una sílaba desde el diálogo de expresiones, deben aparecer las expresiones guardadas de esa sílaba en el cuadro de estado. Por ello, se indica que el cuadro de estado tiene que modificarse. La variable **silb** está declarada como *extern* en *HNS_gestosDlg.h*, y es manejada en el interfaz principal, para saber siempre qué sílaba está seleccionada actualmente.

En la función *OnBorrar()* se trata el borrado de todas las selecciones, tanto las guardadas (las validadas y guardadas se encuentran en el interfaz principal) como las temporales (primer y segundo nivel). Para cada una de estas operaciones, se realiza lo siguiente:

• **Borrado de las selecciones guardadas**: se envía un mensaje para cada sílaba y para cada tipo de expresión existente (3 sílabas x 6 tipos de expresiones), mensaje **10010**, que es donde enviamos los tipos de expresiones, con un 0 en todos los índices, indicando que no hay nada seleccionado. En el momento que algún mensaje **10010** llega al interfaz principal, esto quiere decir que esa selección es válida. Además, se envía un mensaje **10011**, ya que el cuadro de estado ha cambiado, y no debe aparecer ninguna selección. • **Borrado de las selecciones temporales**: se resetea la variable **gestosAct**, poniendo un 0 en todos los índices, esto es, en el campo *lParam*. Además, se notifica a los hijos (diálogos insertados en las pestañas) mediante el mensaje **10005**, para que en los recursos no aparezca ningún elemento seleccionado.

Tabla 24: Esquema de *OnBorrar()*

La función *OnCerrarg()* indica que se ha cerrado el diálogo validando las selecciones. Se realizan tres tipos de acciones:

• Se envía al diálogo *Boca* el mensaje de validación, para que compruebe el campo *PRONUNCIACIÓN DE LABIOS*, que debe predominar sobre las demás selecciones del diálogo. Este mensaje es el **10009**.

• Se envía un mensaje por cada tipo de expresión (6 grupos de expresiones) al interfaz principal indicando las selecciones temporales (que se encuentran en **gestosAct**), para que se conviertan en selecciones guardadas. Para enviar selecciones se utiliza el mensaje **10010**. • Se indica que el diálogo ha sido cerrado mediante el mensaje **10003**. El protocolo de este mensaje sirve para que únicamente pueda haber un diálogo de expresiones abierto.

Tabla 25: Esquema de *OnCerrarg()*

En la función de cerrado del diálogo sin validación, *OnCancelar()*, únicamente se notifica que el diálogo ha sido cerrado (mensaje **10003**), ya que las selecciones temporales no han sido validadas y por tanto no deben llegar al interfaz principal.

En el **tercer nivel**, el correspondiente al interfaz principal (*SeaHamSiGMLDlg.cpp*), únicamente se tratan mensajes entrantes. Estos son los siguientes (únicamente describimos los relacionados con la gestión de expresiones):

- Mensajes entrantes:
    - **10003** → El diálogo de expresiones se ha cerrado, por lo que ya se puede volver a lanzar otro. Esto se indica poniendo **gestosAbierto** a *false*.
    - **10010** → Se han enviado selecciones validadas. Se tratan mediante *ponGesto*, al que se le pasa el *wParam* y el *lParam* enviados, así como **silb**, que se encontraba en *HNS_gestosDlg.h*.
    - **10011** → Se notifica que el cuadro de estado debe modificarse. Se trata mediante la función *actualizaDGestos()*.
    - **10007** → Se debe presentar el vídeo de muestra de una de las expresiones del diálogo *Boca*. Los vídeos tienen el nombre que aparece en los elementos de las listas desplegables, combinando la letra inicial de la lista con el número del elemento (por ejemplo el elemento D02 de la lista *DIENTES*). Para obtener la inicial hacemos uso de la tabla **boca**, obteniendo su posición del 3º y 4º byte de *wParam*, que hemos enviado en el mensaje. Por ello, debemos dividir *wParam* entre 256. El número del elemento seleccionado lo hemos enviado en *lParam*. Para lanzar el programa de vídeo externo, hacemos uso de la función *ShellExecute*.

La función *ponGesto* se encarga de tratar las selecciones realizadas (selecciones guardadas). Realiza dos acciones básicas: guarda las selecciones enviadas e introduce el código SiGML correspondiente a las expresiones en la

codificación actual con la que se está trabajando, variable **signo**. Esta función, además de tratar directamente en la recogida de mensajes **10010**, se llama en la conversión de HamNoSys a SiGML, en *OnHns2sigml()*. De esta manera, para cualquier signo en HamNoSys que editemos, aparecerán las expresiones de la cara especificadas anteriormente. En *OnHns2sigml()* se llama a *ponGesto* tres veces, una por cada sílaba, ya que *ponGesto* únicamente inserta expresiones en una sílaba. La estructuración de operaciones en la función *ponGesto(WPARAM wParam, LPARAM lParam, int sil)* es la siguiente:

- Actualizamos el búfer de gestos guardados con el nuevo recibido
- Copiamos el contenido de **signo** en **buffer**
- Apuntamos el búfer **p** al lugar donde copiar las etiquetas no manuales (1ª sílaba)
- Si la sílaba a insertar es la 2º ó 3º:
    - Apuntamos **p** al siguiente lugar de etiquetas no manual (2ª sílaba)
- Si la sílaba a insertar es la 3ª:
    - Apuntamos **p** al siguiente lugar de etiquetas no manual (3ª sílaba)
- Si **p** no es nulo (existe la sílaba en la que se quieren insertar las expresiones):
    - Copiamos en **copia** el contenido apuntado por **p**
    - Borramos **buffer** a partir de donde ha quedado el apuntamiento de **p**
    - Para cada tipo de expresión (6 grupos de expresiones):
        - Se concatena la etiqueta correspondiente en **buffer**
    - Se concatena **copia** a **buffer**
    - Copiamos en **signo** el contenido de **buffer**, para actualizar **signo**
- Modificamos el cuadro de estado

Tabla 26: Esquema de *ponGesto(…)*

Las expresiones que guardamos quedan reflejadas en el array **gestos**. El tamaño de esta variable es de 3x6x2. El primer índice indica la sílaba, el segundo el tipo de expresión, y el tercero es el que guarda los datos, *wParam* en el 0 y *lParam* en el 1.

Los sucesivos apuntamientos del puntero **p** se realizan para insertar las expresiones en la sílaba correcta, siempre editándolas entre las etiquetas *<hamnosys_nonmanual>* y *</hamnosys_nonmanual>* de la sílaba correspondiente (en la codificación SiGML concatenamos signos, por lo que aquí equivaldría a un signo). La sílaba correspondiente es la última seleccionada en el diálogo de expresiones, que queda guardada en la variable **silb**.

Cuando concatenamos en **buffer** la etiqueta correspondiente a la expresión insertada, hacemos uso de la función *traduceGesto(WPARAM wParam, LPARAM lParam, int sil)*. Esta función se encarga de decodificar el contenido de *wParam* y *lParam* y asignar a la variable global **tagGesto** la etiqueta correspondiente, que en *ponGesto* será insertada en el código SiGML. Utiliza una estructura de tipo *switch*, diferenciando los casos de cada una de las expresiones (diferencia por tanto entre 6 casos). Decodifica *wParam* y *lParam* según la correspondencia de la tabla 1. Una vez obtenido el recurso correspondiente, hace uso de las tablas de expresiones (**boca**, **hombros**, **mirada**, **cejas**, **parpados**, **nariz**, que se encuentran en *Gesto0.cpp*) para editar la correspondiente etiqueta, según el grupo al que corresponde:

1. <hnm_mouthpicture picture=XX/>
2. <hnm_shoulder tag=XX/>
3. <hnm_eyegaze tag=XX/>
4. <hnm_eyebrows tag=XX/>
5. <hnm_eyelids tag=XX/>
6. <hnm_nose tag=XX/>

En el primer caso, cuando la expresión pertenece a *PRONUNCIACIÓN DE LABIOS*, hay que tener en cuenta que en el campo a rellenar en su etiqueta (las XX de <hnm_mouthpicture picture=XX/>), lo que hay que poner es la representación en SAMPA de los alófonos de las palabras a pronunciar por el agente animado. Para obtener esta representación, se hace uso de la función *GraToFon*, desarrollada por el departamento del GTH, y que tiene como objetivo convertir una secuencia de grafemas en la secuencia de alófonos correspondiente en SAMPA. Es necesaria esta conversión porque en la codificación SiGML, la codificación SAMPA es la utilizada para ser posteriormente entendida por VGuido. La secuencia de grafemas es lo que se debe escribir en el cuadro de texto *PRONUNCIACIÓN DE LABIOS*, que hemos guardado en la variable **labios** (esta variable es un array de tres elementos, uno por sílaba).

Al final de la función *ponGesto* se modifica el cuadro de estado. Esto se realiza mediante la función *actualizaDGestos()*. También se llama a esta función cuando se recibe el mensaje **10011**, de modificación de cuadro de estado. Esta función se encarga de modificar el cuadro de texto, mostrando siempre las expresiones guardadas correspondientes a la última sílaba seleccionada en el diálogo de expresiones. El cuadro de estado es un recurso de tipo estático, al que le hemos asociado una variable de tipo *CString*, **m_dispg**.

Para editar esta variable, utilizamos la función *Format*. En ésta vamos insertando cada uno de los campos correspondientes a las expresiones guardadas en **gestos**. Para editar correctamente el tipo de expresión, hacemos

uso de las tablas de expresiones (que se encuentran en *Gesto0.cpp*), para que aparezca el código correspondiente a cada expresión en concreto.

Hay tres tipos de formato en el cuadro de texto (los campos %s corresponden al código de cada expresión):

- Si existe texto editado en *PRONUNCIACIÓN DE LABIOS*:
    - %s-%s-%s-%s-%s-%s-%s → En el primer campo se edita la grafía a pronunciar
- Si no hay ninguna expresión del primer grupo (diálogo *Boca*):
    - No-%s-%s-%s-%s-%s-%s
- En los demás casos:
    - %s-%s-%s-%s-%s-%s-%s → En el primer campo se edita el código correspondiente a una de las listas desplegables del diálogo *Boca*, como por ejemplo “D01”

8.6. Representación animada del signo

El agente animado puede representar siempre que lo deseemos el signo correspondiente al actual con el que estemos trabajando (equivalencia HamNoSys-SiGML). Existen dos formas de activar la representación animada:

- Mediante la pulsación de la tecla ENTER cuando el puntero del ratón se encuentra en el cuadro de edición de HamNoSys
- Mediante la pulsación del botón *Representar*, que se encuentra a la izquierda del cuadro de estado de expresiones

Figura 65: Botón *Representar*

La pulsación de la tecla ENTER la tratamos mediante la sobrescritura de la función *PreTranslateMessage(MSG* pMsg)*. Dentro de esta función, tratamos el mensaje *WM_KEYDOWN* de pulsación de tecla. A su vez, comprobamos si la tecla pulsada ha sido la tecla ENTER, además de comprobar si el foco se encuentra en el cuadro de texto de edición en HamNoSys. En caso afirmativo llamamos a la función *OnPlaysigml()*, que es la encargada de realizar la representación.

El botón *Representar* lo tratamos mediante la función *OnPlaysigml()*, que recoge el mensaje de pulsación de botón *ON_BN_CLICKED*.

La función *OnPlaysigml()* realiza dos operaciones básicas, en este orden:

• Llama a la conversión de HamNoSys a SiGML para actualizar **signo**, ya que la edición de HamNoSys ha podido ser modificada. • Manda al agente animado que reproduzca el signo. Para ello, hace uso de la variable manejadora del control ActiveX VGuido, en este caso la variable **m_avatar**. Dentro de esta variable hay que llamar a la función *PlaySiGML*, que es la encargada de llamar a la reproducción, a la que debemos pasarle el contenido SiGML del signo, que se encuentra en la variable **signo**, contenido SiGML actual con el que estamos trabajando.

Tabla 27: Esquema de *OnPlaysigml(…)*

Figura 66: Fase de la representación de , editado en HamNoSys

9. Generación de la base de datos de signos

En este apartado vamos a describir la generación de la base de datos con los signos necesarios para los dos dominios de aplicación desarrollado: renovación del DNI y del carné de conducir. Comenzaremos describiendo y comparando las tres estrategias de signo-escritura consideradas, después analizaremos el proceso de generación de los signos, y finalmente describiremos el contenido de la base de datos generada.

9.1. Propuestas de signo-escritura

Debido a que la base de la aplicación es la Lengua de Signos Española, más concretamente sus representaciones escritas, comenzaremos con una breve introducción a éstas, centrándonos principalmente en el uso que se hace de ellas en la aplicación.

Los tres tipos de estándares que vamos a detallar son los que más se utilizan en el ámbito de la Lengua de Signos Española, y son los siguientes:

- **GLOSAS**
- **HamNoSys**
- **SEA**

En la aplicación como tal únicamente se utilizan los estándares de glosas y HamNoSys, aunque la representación de signos en SEA está muy extendida en el ámbito de la Lengua de Signos Española, y de hecho ha sido muy útil y necesario a la hora de poder realizar el proyecto. Sería fácilmente integrable.

9.1.1. Glosas

Es la representación más parecida a la lengua escrita castellana, y que podría pensarse que es poco intuitivo para una persona sorda que no conozca esta lengua o tenga dificultades para comprenderla.

En términos generales, una glosa que representa un signo en concreto, es la representación en mayúsculas de la palabra cuyo significado es el que se quiere dar a entender mediante ese signo. Si queremos representar el signo que quiere significar “mesa”, la glosa correspondiente sería MESA. Normalmente la representación de las glosas se hará de esta manera, aunque hay casos en el que el significado que se representa en un signo no tiene su equivalente en castellano con una única palabra. Este sería el caso de la glosa CADA-CINCO-AÑOS, que en castellano habría que especificarlo mediante tres palabras, y en el caso de la Lengua de Signos Española su significado tendría cierta totalidad, por lo que se representa mediante un único signo.

Este tipo de representación es sencilla en cuanto a formato, y fácil de representar en cualquier tipo de documento, aunque, como ya hemos dicho, no siempre es intuitiva, lo que hace más difícil su comprensión.

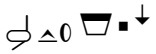
Otro de los problemas que presenta este estándar es su normalización, ya que no es oficial en absoluto, y ciertos aspectos básicos como el guión aparecido en CADA-CINCO-AÑOS no es trivial, y en diferentes documentos podría no utilizarse. Así también ocurre con los caracteres de repetición “++”, que por ejemplo en la glosa CADA-CINCO-AÑOS suele indicarse al final, y con varios caracteres referentes a la unión de ciertos signos o significados.

9.1.2. HamNoSys

Este método ([5]) es uno de los más utilizados a nivel internacional, estando bastante extendido. Se ideó en una universidad alemana a finales de los años ochenta, y desde entonces se ha ido mejorando y dotando de nuevas características. Es un sistema de trascripción fonética en el que en vez de simbolizar los sonidos de una lengua oral, lo que se simboliza en este caso son las formas y los movimientos de las manos. De esta manera se consigue que la comprensión de la lengua de signos sea mucho más sencilla a la hora de leerla en un papel o documento. Este estándar se basa totalmente en la lengua de signos, donde lo importante son esas formas y movimientos de las manos, al contrario que en las glosas, basada en una lengua oral y su simbolización en lengua escrita.

Otra característica importante de este estándar es que la manera de simbolizar estas formas y movimientos no se realiza mediante caracteres, como ocurre en la trascripción fonética de cualquier lengua oral, sino que se consigue mediante símbolos, lo que hace que este estándar de signo-escritura sea mucho más visual e intuitivo. De esta manera, la comprensión de este estándar en comparación con las glosas podría ser más sencilla entre las personas sordas de nacimiento (prelocutivas).

A continuación vamos a detallar de manera muy breve cómo se transcribe un signo cualquiera mediante HamNoSys. El signo que representaría el significado de “persona” se realiza únicamente con una mano, la dominante, con los dedos índice y pulgar extendidos, realizando un pequeño movimiento en línea recta hacia abajo. La trascripción de esta forma de la mano con ese movimiento en HamNoSys sería el siguiente:



Esta trascripción se compone de seis símbolos, aunque realmente se corresponde con siete caracteres (el carácter referente a la posición del pulgar se añade al de la forma de la mano). Obviando el número de caracteres, detallaremos esta trascripción basándonos en los seis símbolos que lo componen y en los cuatro campos en los que se divide cada trascripción de un signo en HamNoSys. Estos cuatro campos son: forma de la mano, orientación de la mano, localización de la mano y movimiento o movimientos de la mano. Con estos cuatro campos nos estamos refiriendo a la mano dominante, aunque en el caso en el que se necesite de la mano pasiva para representar el signo, aparecerán cuatro campos análogos.

La forma de la mano se simboliza en el ejemplo con el primer símbolo. Este representa una mano con el dedo índice extendido, al que se le ha añadido la posición del pulgar, en este caso extendido en perpendicular a la palma. En el campo de la orientación de la mano tenemos dos símbolos en el ejemplo. El primero nos indica la dirección que toma la mano (extendida con la palma hacia abajo), en este caso hacia fuera del cuerpo. El segundo símbolo nos indica la orientación de la palma, hacia la izquierda en el ejemplo. Los dos siguientes símbolos del ejemplo se corresponden con el campo de localización de la mano. El primero de ellos nos indica que la mano debe encontrarse en el tronco en su parte de arriba. El segundo nos indica que la mano no se encuentra centrada en el tronco, sino que está un poco alejada de éste, hacia la derecha. Como el símbolo se encuentra a la derecha (según leemos) del símbolo del tronco, la mano se encontrará en el lado derecho (del signante). Por último, nos queda el símbolo representado con una flecha, que se corresponde con el campo de movimiento o movimientos de la mano. En el ejemplo nos indica un movimiento rectilíneo hacia abajo.

Figura 67: Representación de

Como hemos podido comprobar en el ejemplo, una de las ventajas principales de este estándar es que la trascripción se realiza mediante símbolos visuales, con lo que intuitivamente se consigue una importante comprensión. Otra de las ventajas de este estándar es que es fácilmente generalizable a cualquier lengua de signos, como cualquier sistema de trascripción fonética, lo que no ocurre en las glosas, basadas únicamente en la lengua castellana.

9.1.3. Sistema de Escritura Alfabética (SEA)

Este método ([1]) ha sido desarrollado por una universidad española (Universidad de Alicante), y se trata de un sistema de trascripción fonética, al igual que HamNoSys. La diferencia fundamental entre estos dos estándares es que en HamNoSys se utilizan símbolos visuales a la hora de transcribir, mientras que en SEA se utilizan caracteres corrientes del alfabeto latino con diversos acentos. El objetivo fundamental que se ha buscado al utilizar

caracteres corrientes ha sido el de conseguir mayor normalización y facilidad a la hora de presentar las diferentes trascripciones, principalmente pensando en los formatos electrónicos.

El formato del SEA es muy parecido al de HamNoSys, dividiéndose cada signo en (para un signo de mano dominante, ya que para signos bimanuales habría que añadir un campo de simetría, al igual que ocurre en HamNoSys) seis campos: lugar, contacto, configuración, orientación, dirección y forma. El lugar corresponde al de localización de HamNoSys. El campo de contacto se refiere al posible contacto de la mano con el cuerpo, convirtiéndose en un punto cuando no lo hay. La configuración nos indica la forma de la mano. El campo de orientación cumple la misma función que el correspondiente de HamNoSys, y los campos de dirección y forma se refieren a los movimientos direccionales y formas de movimiento (modificadores de los movimientos, como podrían ser las repeticiones).

Al igual que en HamNoSys, vamos a mostrar un ejemplo de un signo trascrito en SEA, detallando cada uno de los campos de éste. El signo que quiere significar “abrir puerta” se representa con la mano dominante, con el puño cerrado, haciendo un pequeño giro de muñeca de noventa grados, como si estuviéramos abriendo una puerta con una llave. Su representación en SEA es la siguiente:

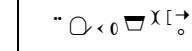
aëmeucre

El primer campo correspondería al lugar de localización de la mano, aunque en este caso no aparece, lo que nos indica que debemos utilizar el de por defecto, la mitad del tronco en el lado de la mano con la que se representa el signo (en este caso la mano dominante). El segundo campo es el del contacto con el cuerpo. En este caso no importa, ya que utilizamos el lugar por defecto, en el lado de la mano dominante sin pegarse al tronco. El siguiente campo es el de la configuración de la mano, siendo en este caso “aë”. Esta configuración es la del puño cerrado, con el dedo índice un poco más levantado, aunque también flexionado, envolviendo la punta del dedo pulgar (que se encuentra estirado). En el campo de la orientación nos encontramos “meu”. Esto nos indica que la orientación es “me”, que sería la orientación natural de la mano, aunque modificada, por eso aparece la “u”. La orientación natural “me” correspondería con la proyección “natural” de la mano sobre el antebrazo estirado en perpendicular al tronco (posición por defecto), lo que en este caso sería una dirección hacia fuera del cuerpo y la palma hacia la izquierda (suponiendo la derecha como mano dominante). La modificación de la “u” nos indica que la palma mira hacia abajo, en vez la posición natural “me”. Por último, nos quedarían los campos de dirección y formas de movimiento. En este caso tenemos “cre”, que nos indica un movimiento de giro de muñeca de noventa grados en sentido horario (desde la visión del signante).

Figura 68: Representación de *aëmeucre*

Como vemos, el sistema es muy parecido al de HamNoSys, con la diferencia de la utilización de caracteres alfabéticos. Como hemos visto, esto tiene sus ventajas a la hora de su presentación y distribución en formatos electrónicos, además de la posibilidad de utilizar teclados corrientes para transcribir cualquier signo, aunque también tiene sus desventajas. Al tratarse de caracteres corrientes, a diferencia de los símbolos visuales de HamNoSys, hace que este estándar sea mucho menos intuitivo, y haya que conocer en cierta profundidad el SEA para poder comprender la trascripción. Para una persona que nunca haya visto nada de SEA la comprensión de un signo se hace bastante complicada. Otra de las desventajas de este estándar es que se ha diseñado expresamente para la Lengua de Signos Española, con lo que hay formas o movimientos que encontraríamos en otras lenguas de signos que no se especifican en este estándar. Esto hace que sea menos generalizable que HamNoSys, que tiene muchas más posibilidades, además de estar mucho más extendido a nivel internacional, a diferencia del SEA que se reduce al ámbito español.

A continuación vamos a presentar una pequeña tabla resumen en la que presentamos las características de estos tres estándares:

	GLOSAS	**SEA**	**HamNoSys**
Ejemplo: “cualquiera”	CUALQUIERA	s ymòazy-faho	
Tipo de estándar	Se representa el significado del signo representado mediante su equivalente en castellano	Sistema de trascripción fonética, trascribiendo formas y movimientos de las manos, no sonidos	Sistema de trascripción fonética, trascribiendo formas y movimientos de las manos, no sonidos

	(mayúsculas)		
Tipo de fuente	Caracteres corrientes (castellano)	Caracteres corrientes y diversos acentos	Caracteres simbólicos
Ventajas	Equivale a su significado. Normalización de formato y posibilidad de utilizar teclado corriente	Fácil comprensión por personas sordas por representar formas y movimientos. Normalización de formato y posibilidad de utilizar teclado corriente	Fácil comprensión por personas sordas por representar formas y movimientos. Muy generalizable. Internacionalización. Codificación por símbolos intuitiva.
Desventajas	Difícil comprensión para personas sordas que no dominan el castellano	Codificación de formas y movimientos poco intuitiva. No se contemplan todas las formas y movimientos	Necesidad de fuentes especiales para su codificación. No se pueden utilizar teclados corrientes de una manera sencilla e intuitiva

Tabla 28: Comparación entre tipos de representación escrita de signos

9.2. Proceso de generación de los signos

La generación de los signos (tanto signos generales como signos incluidos en la base de datos de ejemplos) ha sido llevada a cabo mediante la visión de videos en los que eran representados los signos. Tres han sido las fuentes principales de las que hemos obtenido estos videos:

- Diccionario normativo de la Lengua de Signos Española ([6])
- Diccionario de Lengua de Signos Española ([16])
- Grabación de ejemplos por la Fundación CNSE

La mayoría de los signos han sido generados a partir de los videos incluidos en el diccionario normativo, ya que, además de los propios videos se incluía para cada signo su representación escrita en SEA. Este tipo de escritura es la más extendida en el ámbito de la Lengua de Signos Española, siendo la más conocida y tratada por la Fundación CNSE, autora del diccionario normativo.

Como ya hemos comentado, la base de datos necesita la representación en HamNoSys de cada signo (el archivo SiGML correspondiente es equivalente), ya que es el único lenguaje que acepta el agente animado. La generación de cada signo, por tanto, debería ser realizada codificando la representación del

signo (a partir de la visión del video correspondiente) en HamNoSys, a partir de las formas y movimientos observados. Sin embargo, el hecho de tener una gran base de datos (el diccionario normativo) con la representación de cada signo en SEA incluida, podía ser algo muy aprovechable, lo que desembocó en el diseño e implementación del editor SEA-HamNoSys (que se detalla en los capítulos correspondientes).

Este editor tiene el objetivo de intentar realizar una conversión entre SEA y HamNoSys, intentando ser fiel en la descripción de configuraciones y movimientos. De esta manera se conseguiría obtener de una forma mucho más rápida la base de datos con los signos representados en HamNoSys, a partir de la del diccionario normativo, con su representación en SEA. La motivación de realizar este conversor se basa en la similitud de estructuras entre los dos formatos, ya que los dos tratan de transcribir los mismos campos de descripción de la representación del signo: configuración, dirección, localización y movimientos.

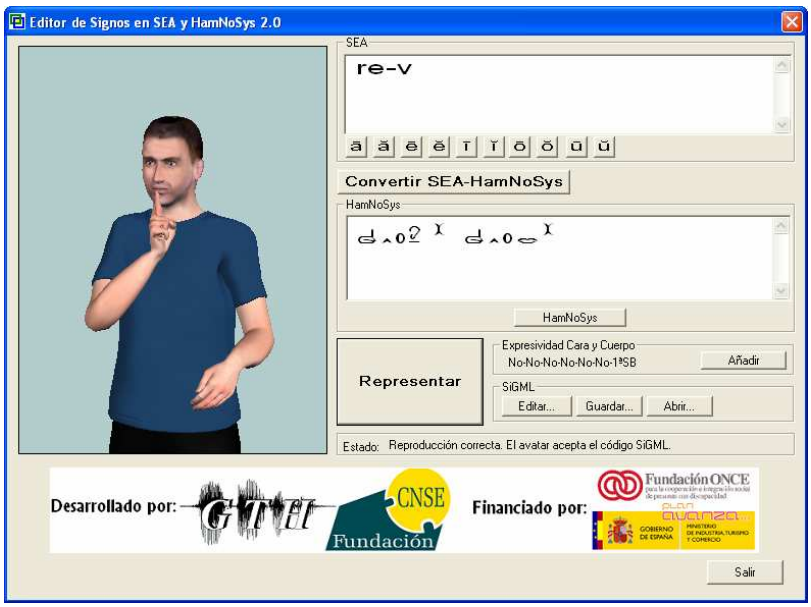


Figura 69: Editor de Signos en SEA y HamNoSys

El conversor no puede realizar la totalidad de conversiones, ya que la codificación de cada formato tiene ciertas diferencias, además de las excepciones del propio SEA, siendo muy difícil la conversión perfecta. Las pequeñas diferencias en las codificaciones de los formatos generan grandes problemas en la conversión, pero además hay que tener en cuenta que la representación de los signos se lleva a cabo por el agente animado a partir de HamNoSys, existiendo también ciertas peculiaridades y excepciones. Por otro lado, la representación natural de los signos hace que cada representación del mismo signo, por distintos signantes o por el mismo signante en diferentes situaciones, sea diferente, con lo que subjetivamente las representaciones del agente animado siempre puedan tener pequeñas modificaciones. Los cambios actuales que se están produciendo en la notificación del SEA hacen surgir más

problemas en la conversión, lo que hace que periódicamente haya que realizar cambios en la misma.

Sin embargo, en la mayoría de los casos (aproximadamente un 70%) la conversión es bastante buena (quizá requieran de algún retoque subjetivo, dadas las limitaciones del agente animado y las limitaciones de las propias especificaciones de los dos formatos de signo-escritura). En los casos restantes se encuentran un grupo de signos en los que la conversión no genera un signo correcto completamente, pero aún así siempre se genera una especie de plantilla del signo en HamNoSys, a la que sólo habría que cambiar ciertos campos manualmente (retocar algunos de los símbolos de HamNoSys). En algunos casos las limitaciones son demasiado importantes y la conversión necesita grandes modificaciones, aunque son casos muy concretos. Por ello, la utilización de este conversor ha conseguido acelerar y facilitar enormemente la generación de esta base de datos, siendo un elemento fundamental en la realización del proyecto.

Para comprender mejor la generación de signos mediante el conversor vamos a detallar unos ejemplos. Como hemos comentado, existen varios signos tipo en cuanto a la generación de los mismos mediante el conversor, por lo que pondremos un ejemplo de generación de cada uno de estos conjuntos de signos:

- Signos en los que la conversión es prácticamente perfecta (un 70% de los casos mediante el conversor): *ANCHO*

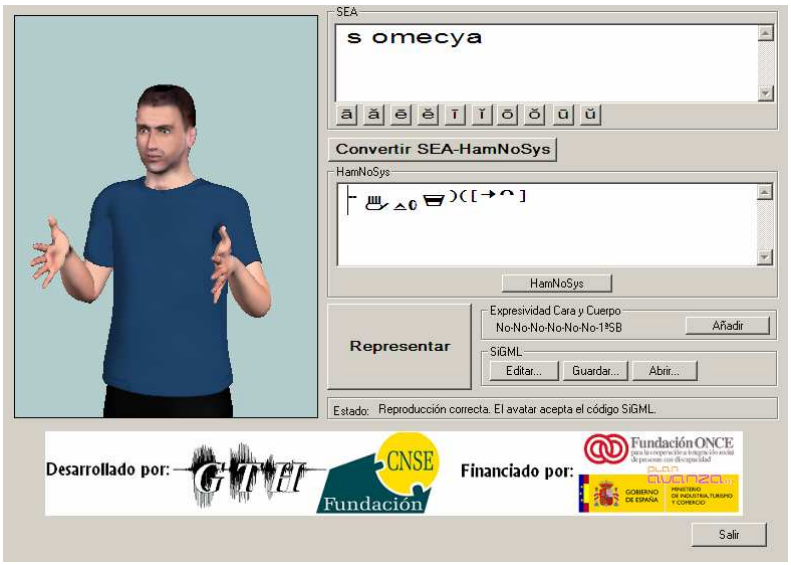


Figura 70: Conversión del signo *ANCHO* a HamNoSys

En este caso no ha sido necesario introducir ningún cambio en la conversión, ya que la trascripción del signo es similar en uno y otro formato de escritura.

- Signos casi perfectos, pero en los que es necesario algún cambio, en principio subjetivo (teniendo en cuenta la visión del video de

representación del signo real). Este cambio es necesario ya sea por una mala conversión o por una codificación no del todo correcta del signo en SEA. Tenemos el ejemplo del signo: *GANA*

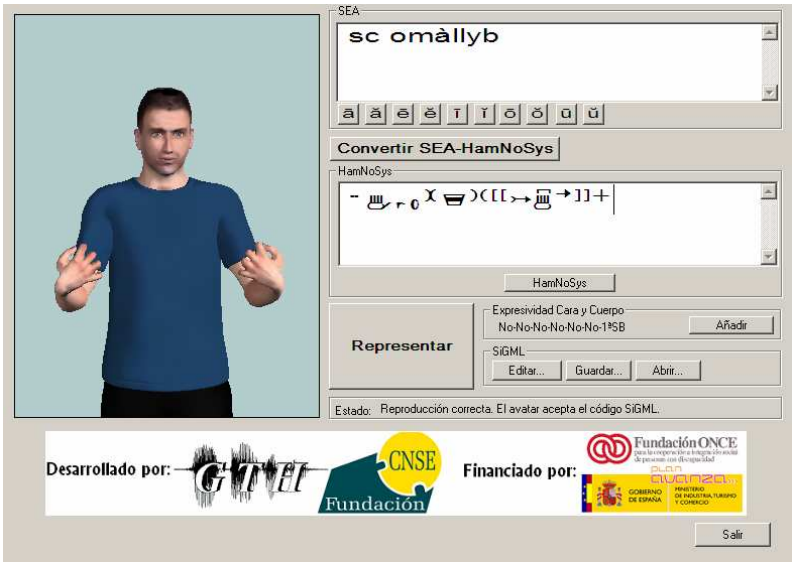


Figura 71: Conversión del signo *GANA* a HamNoSys

En este caso es prácticamente perfecta, aunque es necesario realizar pequeñas modificaciones. Una de ellas es el cambio de configuración, ya que en la representación real del signo las palmas de las manos están prácticamente enfrentadas. Esto puede ser debido a varias causas. Una de ellas podría ser que en la representación del signo la configuración del signo no estuviera del todo unificada, o que simplemente la propia repetición y aceleración de la mima representación pudieran llevar a esta diferencia. Otra posibilidad es que no se haya codificado correctamente el signo en SEA, habiendo un error en la configuración. Por ello habría que realizar este pequeño cambio, dejando la configuración en:

Las otras modificaciones son totalmente subjetivas. La primera de ellas sería colocar las manos un poco más alejadas del cuerpo (para ello habría que eliminar el carácter  que aparece a continuación de la localización). La otra modificación subjetiva corresponde a conseguir que el movimiento de separación de las manos sea un poco más corto, sustituyendo el movimiento rectilíneo por:

De esta manera, teniendo en cuenta las tres pequeñas modificaciones realizadas, la codificación en HamNoSys final de este signo sería la siguiente (muy similar a la que se obtuvo de la conversión SEA-HamNoSys):

- Signos en los que siempre es necesario realizar alguna modificación, aunque siempre se presenta una plantilla del signo a codificar. Este sería el caso del signo: *ADHESIÓN*

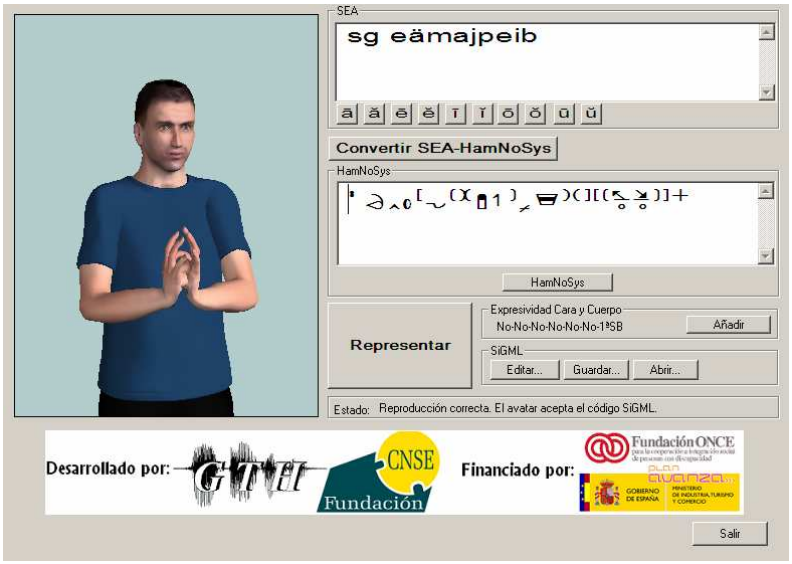
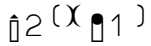


Figura 72: Conversión del signo *ADHESIÓN* a HamNoSys

La estructura del signo que proporciona el conversor es la correcta, aunque habría que precisar el enlazado de las manos, para que pareciera más natural y cercano al de la representación real. Para ello podríamos retocar la configuración de las manos y la localización. En cuanto a la primera, el conversor ha decidido una configuración simétrica en ambas manos, pero se podría retocar la de la mano no dominante (la izquierda en el agente animado). El enlazado sería más correcto si las direcciones de ambas manos difirieran en noventa grados, en vez de estar totalmente enfrentadas. Así, la configuración más correcta de la mano no dominante sería:

El principal cambio habría que realizarlo en la localización, para indicar un contacto correcto entre las dos manos. Para ello se modifica el contacto de la mano dominante, indicando el contacto entre dedos de las manos, para conseguir el enlazado correcto. Éste es uno de los inconvenientes de la especificación del SEA, en la que no se contemplan diferencias en los contactos entre las manos, ni se pueden detallar con exactitud, por lo que al convertirlo en HamNoSys es totalmente necesario realizar un retoque manual en este sentido. En el caso del ejemplo habría que indicar contacto entre la punta del dedo índice de la mano no dominante con la yema del dedo pulgar de la mano dominante, cuya codificación en HamNoSys sería la siguiente:



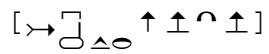
La codificación final del signo en HamNoSys, habiendo precisado el enlazado de las manos, quedaría de la siguiente forma:

En todo caso, a pesar de haber tenido que realizar ciertas modificaciones para conseguir una representación más fiel del signo por parte del agente animado, se presenta una plantilla del signo, ya que tenemos la estructura correctamente indicada, en este caso configuración y localización en ambas manos, quedando sólo que modificar los campos dentro de la estructura.

- Signos, que debido a las limitaciones de las especificaciones en la escritura y en el agente animado, no obtienen una conversión correcta, habiendo que retocar bastante ciertas partes de la codificación del signo. Este sería el caso del signo: *NATURAL*

Figura 73: Conversión del signo *NATURAL* a HamNoSys

En este caso concreto la limitación viene impuesta por la especificación del movimiento en SEA. En ésta se indica un movimiento circular desde la muñeca (*wrahe*), sin llegar a detallar el movimiento más. Esto hace que al convertirlo a HamNoSys haya que detallar mucho más el movimiento, ya que no es suficiente con un simple movimiento circular de la mano desde la muñeca, ya que habrá que mover la mano en un pequeño movimiento circular absoluto (no desde la muñeca), a la vez que cambiamos la configuración de la mano para dar esa sensación de giro de muñeca. Así, habría que codificar manualmente el signo, visualizando las grabaciones reales de la representación del signo. El movimiento codificado manualmente quedaría de la siguiente manera:



De esta manera se han ido generando la mayoría de los signos de la base de datos, obteniendo las codificaciones en HamNoSys a partir de la base de

datos del diccionario normativo (con la descripción en SEA) y el conversor. Como hemos comentado, en algunos casos concretos ha habido que realizar modificaciones manuales, fijándonos en las grabaciones reales, pero en la gran mayoría las conversiones han sido muy satisfactorias gracias al editor de SEA.

Algunos de los signos necesarios en la aplicación no se encontraban en el diccionario normativo, por lo que no hemos podido partir del SEA para obtener la codificación en HamNoSys. Estos signos han tenido que ser generados manualmente mirando los videos o las imágenes del Diccionario de lengua de signos. En concreto, los signos de las letras del abecedario fueron generados a partir de este último, a partir de imágenes. La mayoría de los demás signos sin codificación en SEA fueron generados manualmente a partir de los videos grabados por la Fundación CNSE.

La generación del código SiGML (la única que acepta el agente animado para la representación de signos) también la realiza el propio editor, a partir de la codificación HamNoSys insertada. De esta manera, el código SiGML de la totalidad de los signos de esta base de datos, que también debe guardarse en la base de datos, ha sido generado a partir del editor.

La generación del código SiGML permite la introducción de ciertas etiquetas indicadoras de gestos de la cara y del cuerpo. Estas etiquetas también han sido adaptadas al editor, para facilitar su inserción. A la vez que se ha ido generando el código SiGML a partir del editor, se han ido insertando ciertos gestos en algunos de los signos, principalmente la pronunciación del signo correspondiente mediante los labios del agente animado. Esto ayuda enormemente a la comprensión de ciertos signos por parte de las personas sordas, siendo por tanto la lectura de los labios un elemento clave.

9.3. Signos incluidos en la base de datos

Una vez detallado el proceso de generación de la base de datos, queda describir ésta, indicando el número de signos generados, así como su orden y el tipo de datos que acompañan a cada uno.

Esta base de datos contiene todos los signos que podrán ser elegidos desde la interfaz gráfica de la aplicación, para su posterior traducción. Los signos disponibles en el sistema de traducción se presentan en dos formatos distintos de escritura, en forma de glosas y representados mediante HamNoSys. Sin embargo, la mejor manera de presentar un signo de la Lengua de Signos Española es la visual, que es la forma natural de comunicación mediante esta lengua. Por eso, la presentación de cada uno de los signos que están disponibles es fundamental, ya que ayuda enormemente a una elección correcta.

Si únicamente fuera necesaria la presentación en forma de glosas de los signos, la base de datos sería muy sencilla, ya que esta información es similar a la ofrecida por la presentación de la lengua escrita castellana. De hecho, la presentación en glosas, además de ayudar a la búsqueda de signos en la interfaz, tiene un papel muy importante de etiquetado, ya que en la mayoría de

los casos, a nivel de programación en la aplicación se utilizará la presentación en glosas, mucho más clara y fácil de manejar en este caso. Por ello este tipo de información es también fundamental, pero es sencilla de obtener, pudiendo generar rápidamente una base de datos con un gran número de signos.

Los signos disponibles en la interfaz gráfica están representados también en HamNoSys, por lo que también es fundamental guardar esta información, para cada uno de los signos. La información fundamental que queda es la presentación visual de los signos, que se realiza por el agente animado. Esta información que se le pasa al agente animado se codifica mediante un archivo SiGML, siendo el archivo de texto correspondiente el que también se guarde en la base de datos.

Los tres campos fundamentales que deben contener todos los signos de la base de datos son, por tanto, su glosa, su representación en HamNoSys, y su codificación en SiGML (que se guarda en un archivo de tipo texto) correspondiente para poder ser representado por el agente animado. La representación en HamNoSys y las etiquetas incluidas en el archivo de SiGML están íntimamente ligadas, representando cada etiqueta un único carácter de la secuencia de HamNoSys.

Como hemos comentado en el apartado anterior, las oraciones de la base de datos de ejemplos deben cubrir la mayoría de las expresiones y frases intercambiadas entre funcionario y usuario. Debido a este hecho, el conjunto básico de frases que van a ser utilizadas por el sistema de traducción serán las incluidas en la base de datos, por lo que cada uno de los signos que aparece en ésta (en el campo *LSE*) debe estar incluido en la base de datos de signos. De esta manera podrán ser elegidos todos los signos con los que se forman los ejemplos, cubriendo la gran mayoría de casos en una conversación típica entre funcionario y usuario.

El conjunto de signos que aparece en la base de datos de ejemplos será básico, aunque puede haber lugar a cierto número de signos que no aparezcan en los ejemplos, y que pudiera pensarse que sería útil tenerlos disponibles en el sistema de traducción. Estos signos son los agrupados en categorías (números, días de la semana, meses…), ya que basta con uno por categoría que aparezca en algún ejemplo, cubriendo así ese tipo de expresión, aunque cada categoría contenga muchas glosas diferentes.

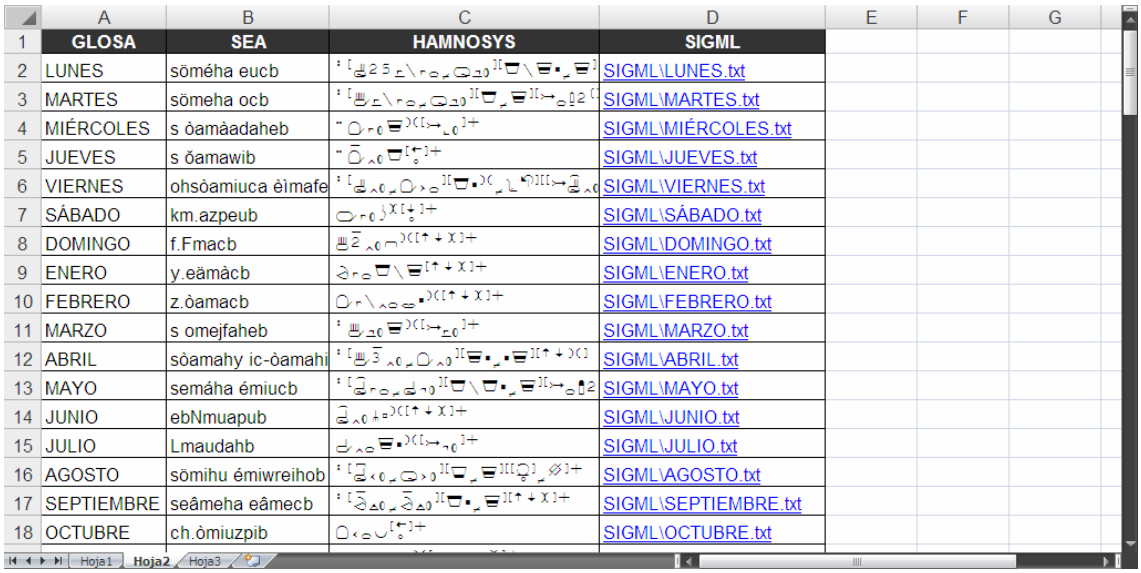
Esta base de datos se compone de 5 conjuntos importantes de signos:

- Signos incluidos en la base de datos de ejemplos de traducción: tanto para el dominio de renovación del DNI como del carné de conducir.
- Signos generales, de uso común, muchos de ellos agrupados en categorías. Dentro de este grupo encontramos otros tres, que son:
    - Días de la semana y meses
    - Letras del abecedario

- Números, del 0 al 100
- Signos para especificar las horas: números del 1 al 12, número cada 5 minutos y expresiones como Y-MEDIA, Y-CUARTO, MENOS-CUARTO, DE-LA-TARDE y DE-LA-NOCHE.

La base de datos se divide en cuatro conjuntos de signos. Cada conjunto se especifica en un fichero de tipo *Excel*.

- Signos que representan los días y los meses, en el fichero *días_meses.xls*

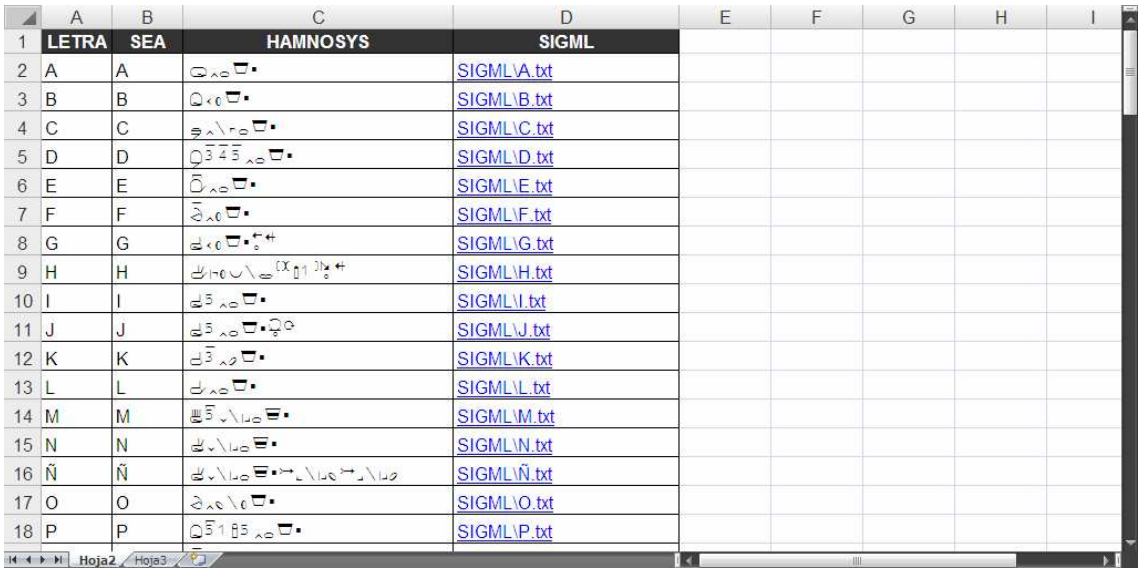


	A	B	C	D
1	GLOSA	SEA	HAMNOSYS	SIGML
2	LUNES	sömèha eucb		SIGML\LUNES.txt
3	MARTES	sömeha ocb		SIGML\MARTES.txt
4	MIÉRCOLES	s òamàadaheb		SIGML\MIÉRCOLES.txt
5	JUEVES	s ðamawib		SIGML\JUEVES.txt
6	VIERNES	ohsòamiuca èìmafe		SIGML\VIERNES.txt
7	SÁBADO	km.azpeub		SIGML\SÁBADO.txt
8	DOMINGO	f.Fmacb		SIGML\DOMINGO.txt
9	ENERO	y.eämàcb		SIGML\ENERO.txt
10	FEBRERO	z.òamacb		SIGML\FEBRERO.txt
11	MARZO	s omejfaheb		SIGML\MARZO.txt
12	ABRIL	sòamahy ic-òamahi		SIGML\ABRIL.txt
13	MAYO	semáha émiucb		SIGML\MAYO.txt
14	JUNIO	ebNmuapub		SIGML\JUNIO.txt
15	JULIO	Lmaudahb		SIGML\JULIO.txt
16	AGOSTO	sömihu émiwreihob		SIGML\AGOSTO.txt
17	SEPTIEMBRE	seâmeha eâmecb		SIGML\SEPTIEMBRE.txt
18	OCTUBRE	ch.òmiuzpib		SIGML\OCTUBRE.txt

Hoja1 Hoja2 Hoja3

Figura 74: Base de datos de signos, *dias_meses.xls*

- Signos que representan las letras del abecedario, en el fichero *letras.xls*

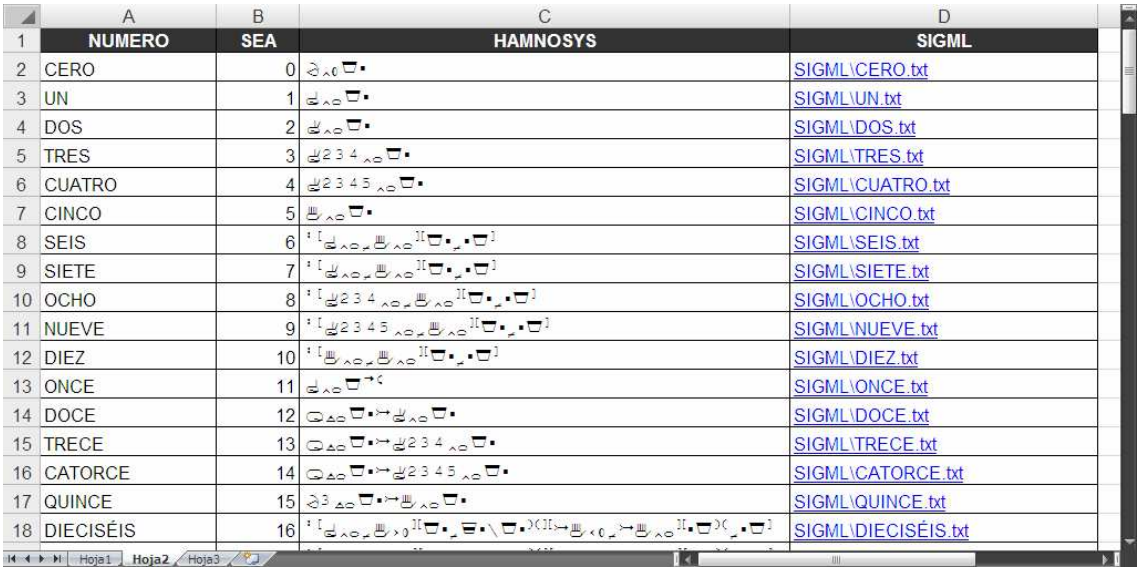


	A	B	C	D
1	LETRA	SEA	HAMNOSYS	SIGML
2	A	A		SIGML\A.txt
3	B	B		SIGML\B.txt
4	C	C		SIGML\C.txt
5	D	D		SIGML\D.txt
6	E	E		SIGML\E.txt
7	F	F		SIGML\F.txt
8	G	G		SIGML\G.txt
9	H	H		SIGML\H.txt
10	I	I		SIGML\I.txt
11	J	J		SIGML\J.txt
12	K	K		SIGML\K.txt
13	L	L		SIGML\L.txt
14	M	M		SIGML\M.txt
15	N	N		SIGML\N.txt
16	Ñ	Ñ		SIGML\Ñ.txt
17	O	O		SIGML\O.txt
18	P	P		SIGML\P.txt

Hoja2 Hoja3

Figura 75: Base de datos de signos, *letras.xls*

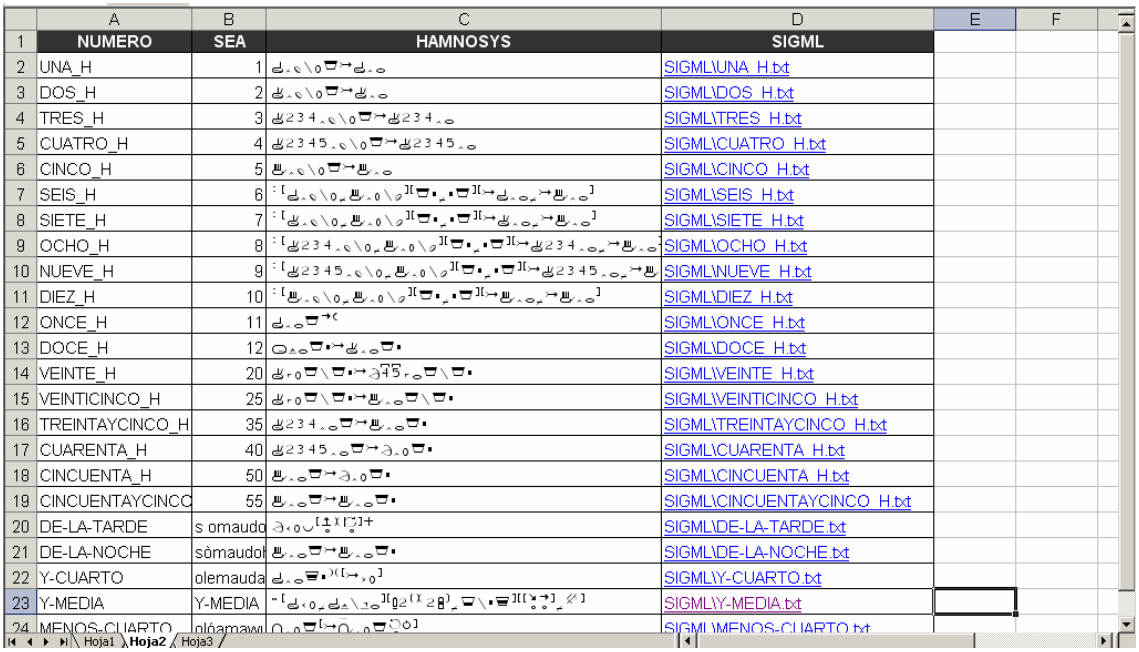
- Signos que representan los números del 0 al 100, 1000 y 1000000, en el fichero *números_signos.xls*



	A	B	C	D
1	NUMERO	SEA	HAMNOSYS	SIGML
2	CERO	0		SIGML\CERO.txt
3	UN	1		SIGML\UN.txt
4	DOS	2		SIGML\DOS.txt
5	TRES	3		SIGML\TRES.txt
6	CUATRO	4		SIGML\CUATRO.txt
7	CINCO	5		SIGML\CINCO.txt
8	SEIS	6		SIGML\SEIS.txt
9	SIETE	7		SIGML\SIETE.txt
10	OCHO	8		SIGML\OCHO.txt
11	NUEVE	9		SIGML\NUEVE.txt
12	DIEZ	10		SIGML\DIEZ.txt
13	ONCE	11		SIGML\ONCE.txt
14	DOCE	12		SIGML\DOCE.txt
15	TRECE	13		SIGML\TRECE.txt
16	CATORCE	14		SIGML\CATORCE.txt
17	QUINCE	15		SIGML\QUINCE.txt
18	DIECISÉIS	16		SIGML\DIECISÉIS.txt

Figura 76: Base de datos de signos, *números_signos.xls*

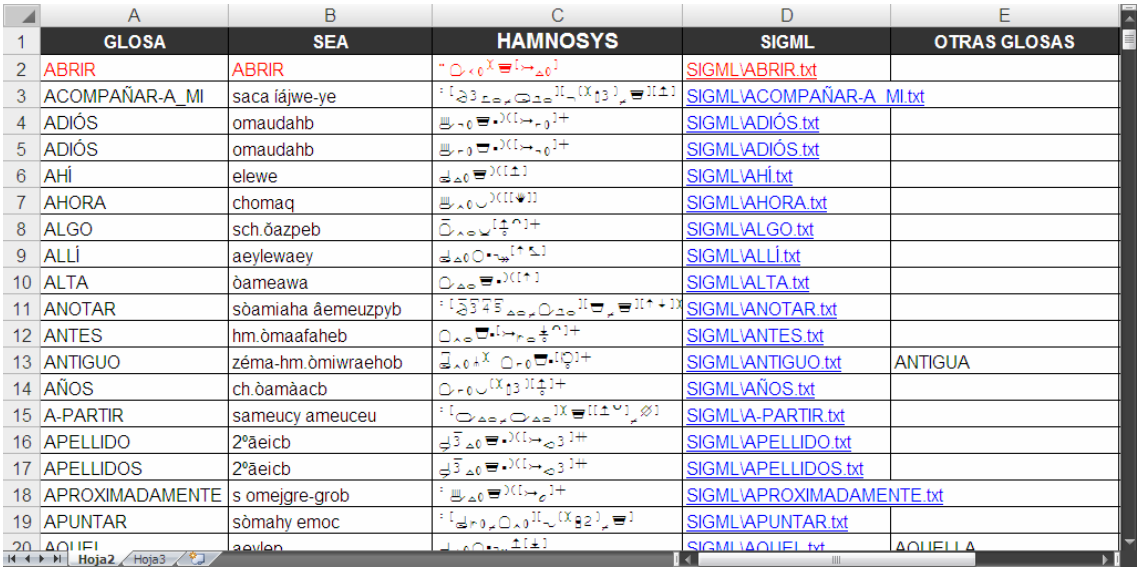
- Signos para especificar las horas: números del 1 al 12, número cada 5 minutos y expresiones como Y-MEDIA, Y-CUARTO, MENOS-CUARTO, DE-LA-TARDE y DE-LA-NOCHE.



	A	B	C	D	E	F
1	NUMERO	SEA	HAMNOSYS	SIGML		
2	UNA_H	1		SIGML\UNA_H.txt		
3	DOS_H	2		SIGML\DOS_H.txt		
4	TRES_H	3		SIGML\TRES_H.txt		
5	CUATRO_H	4		SIGML\CUATRO_H.txt		
6	CINCO_H	5		SIGML\CINCO_H.txt		
7	SEIS_H	6		SIGML\SEIS_H.txt		
8	SIETE_H	7		SIGML\SIETE_H.txt		
9	OCHO_H	8		SIGML\OCHO_H.txt		
10	NUEVE_H	9		SIGML\NUEVE_H.txt		
11	DIEZ_H	10		SIGML\DIEZ_H.txt		
12	ONCE_H	11		SIGML\ONCE_H.txt		
13	DOCE_H	12		SIGML\DOCE_H.txt		
14	VEINTE_H	20		SIGML\VEINTE_H.txt		
15	VEINTICINCO_H	25		SIGML\VEINTICINCO_H.txt		
16	TREINTAYCINCO_H	35		SIGML\TREINTAYCINCO_H.txt		
17	CUARENTA_H	40		SIGML\CUARENTA_H.txt		
18	CINCUENTA_H	50		SIGML\CINCUENTA_H.txt		
19	CINCUENTAYCINCO	55		SIGML\CINCUENTAYCINCO_H.txt		
20	DE-LA-TARDE	s omaudo		SIGML\DE-LA-TARDE.txt		
21	DE-LA-NOCHE	sòmaudol		SIGML\DE-LA-NOCHE.txt		
22	Y-CUARTO	olemauda		SIGML\Y-CUARTO.txt		
23	Y-MEDIA	Y-MEDIA		SIGML\Y-MEDIA.txt		
24	MENOS-CUARTO	olóamawı		SIGML\MENOS-CUARTO.txt		

Figura 77: Base de datos de números para las horas, *horas_signos.xls*

- Signos utilizados en la renovación del DNI y del carné de conducir, que componen la mayoría de la base de datos, en el fichero *dni_dgt.xls.*



	A	B	C	D	E
1	GLOSA	SEA	HAMNOSYS	SIGML	OTRAS GLOSAS
2	ABRIR	ABRIR		SIGML\ABRIR.txt	
3	ACOMPAÑAR-A_MI	saca íájwe-ye		SIGML\ACOMPAÑAR-A_MI.txt	
4	ADIÓS	omaudahb		SIGML\ADIÓS.txt	
5	ADIÓS	omaudahb		SIGML\ADIÓS.txt	
6	AHÍ	elewe		SIGML\AHÍ.txt	
7	AHORA	chomaq		SIGML\AHORA.txt	
8	ALGO	sch.ŏazpeb		SIGML\ALGO.txt	
9	ALLÍ	aeylewaey		SIGML\ALLÍ.txt	
10	ALTA	òameawa		SIGML\ALTA.txt	
11	ANOTAR	sòamiaha âemeuzpyb		SIGML\ANOTAR.txt	
12	ANTES	hm.òmaafaheb		SIGML\ANTES.txt	
13	ANTIGUO	zéma-hm.òmiwraehob		SIGML\ANTIGUO.txt	ANTIGUA
14	AÑOS	ch.òamàacb		SIGML\AÑOS.txt	
15	A-PARTIR	sameucy ameuceu		SIGML\A-PARTIR.txt	
16	APELLIDO	2ºâeicb		SIGML\APELLIDO.txt	
17	APELLIDOS	2ºâeicb		SIGML\APELLIDOS.txt	
18	APROXIMADAMENTE	s omejgre-grob		SIGML\APROXIMADAMENTE.txt	
19	APUNTAR	sòmahy emoc		SIGML\APUNTAR.txt	
20	AQUEL	aeylep		SIGML\AQUEL.txt	AQUELLA

Hoja2 Hoja3

Figura 78: Base de datos de signos, *dni_dgt.xls*

La estructura de cada fichero de *Excel* es la misma en los cuatro casos, por lo que únicamente vamos a describir el formato de uno de ellos, en este caso el fichero *dni_dgt.xls*.

El mayor conjunto de signos que componen esta base de datos es el de los signos incluidos en la base de datos de ejemplos, ya que es necesario que aparezcan todos ellos. Los signos generales son muy importantes, y más teniendo en cuenta el ámbito de trabajo de la aplicación, que se centra en trámites burocráticos relacionados con el DNI y el carné de conducir, donde la indicación de números, letras y fechas es vital.

Como podemos comprobar en la figura, cada fichero se compone de cinco campos:

- **GLOSA:** en este campo se especifica la glosa con la que se etiqueta el signo correspondiente.
- **SEA:** aquí se especifica la trascripción en SEA del signo. Ya hemos comentado que esta trascripción la hemos obtenido de la base de datos del diccionario normativo. Algunos de los signos no aparecían en esta base, por lo que no hemos podido acceder a esta trascripción. En estos casos toda la fila se encuentra pintada en rojo, así como en este campo aparece otra vez la glosa correspondiente, que aparecía en el campo anterior.
- **HAMNOSYS:** en este campo se especifica la codificación en HamNoSys, cuya generación hemos descrito en el apartado anterior.
- **SIGML:** en este campo se indica un enlace al archivo de texto en el que se encuentra el código SiGML, cuyas etiquetas equivalen una a una a cada uno de los caracteres especificados en el campo correspondiente a HamNoSys.

- **OTRAS GLOSAS:** este campo no está relleno en todas los signos, únicamente tiene sentido para aquellos en los que existe más de una glosa con la que se etiqueta la representación de un signo (un único significado).

A continuación vamos a detallar el número de signos que componen cada uno de los cuatro ficheros.

- Fichero *dni_dgt.xls*: se compone de un total de 574 signos, todos ellos necesarios en el ámbito de trabajo del DNI y del carné de conducir. La mayoría de estos signos corresponden a los incluidos en la base de datos de ejemplos.

- Fichero *dias_meses.xls*: se compone de un total de 19 signos, 12 de ellos correspondientes a los meses del año, y los otros 7 representando los días de la semana.

- Fichero *letras.xls*: se compone de un total de 27 signos, correspondientes a las 27 letras del abecedario.

- Fichero *numeros_signos.xls*: se compone de un total de 103 signos, correspondientes a los números del 0 a 100, además del número 1000 y el 1000000.

De esta base de datos el sistema de traducción de voz a LSE únicamente utiliza los ficheros de texto con el código SiGML, ya que es la parte necesaria para que el agente animado pueda representar la totalidad del signo. La conversión en HamNoSys (conversión sencilla debido a la equivalencia entre caracteres de HamNoSys y etiquetas SiGML) se realiza dentro de la propia aplicación.

En total, el número de signos (número de ficheros de texto con el código SiGML) utilizados en la aplicación asciende a 823, sumando los correspondientes a la base de datos previa, además de los especificados en estos cuatro ficheros. Hay que tener en cuenta que muchos de los signos se etiquetan con más de una glosa (lo que especificamos en el quinto campo de los ficheros *Excel*), por lo que algunos de los ficheros de texto con el código SiGML se repiten.

10. Evaluación con usuarios y discusión de los problemas encontrados

En este informe se pretende recoger las principales características de la evaluación del sistema desarrollado para la traducción de voz a Lengua de Signos Española (LSE) aplicado al dominio de la renovación del carné de conducir. En primer lugar se describe el procedimiento de renovación con los papeles que son necesarios. Posteriormente se describe el proceso de evaluación, los escenarios considerados con ejemplos de diálogos, las medidas objetivas y subjetivas resultado de la evaluación, y finalmente, un resumen de resultados.

10.1. Procedimiento de renovación del carné de conducir

Nos vamos a centrar en la renovación del carné de conducir tipo B2 (coches) por ser el más común. Este carné hay que renovarlo cada 10 años hasta los 45, de 45 a 70 cada 5 años, y de 70 años en adelante cada 2 años.

La renovación del carné tiene como motivo más frecuente su caducidad. En este caso la documentación que hay que presentar es la siguiente:

- Impreso de solicitud que nos dan en la oficina de la Dirección General de Tráfico (DGT).
- El carné antiguo.
- El certificado médico (informe de aptitudes psicofísicas)
- DNI (o permiso de residencia)
- Una fotografía.

En el caso de que la renovación se realice por cambio de domicilio será necesario presentar un certificado de empadronamiento en el nuevo domicilio.

En todos los casos hay que pagar una tasa que ronda los 10-15 euros

En la oficina de la DGT de Toledo el procedimiento es el siguiente:

1. En la ventanilla de información nos dan el impreso de solicitud que debemos rellenar.
2. Posteriormente debemos pasar por caja para pagar la oportuna tasa.
3. Finalmente, con el justificante de haber pagado la tasa, el informe y el resto de documentación se pasa a una tercera ventanilla donde se deposita toda la documentación. En esta ventanilla te dan un carné provisional (con validez de tres meses) hasta que te llegue por correo el nuevo carné definitivo.

Para las tres ventanillas (información, caja y conductores) es necesario coger un número y guardar cola. En nuestro caso vamos a suponer que todas las etapas se realizan en un mismo puesto: en una mesa de despacho donde el usuario es atendido por el mismo funcionario para todos los pasos. En la siguiente figura podemos observar la preparación de la mesa para la evaluación:

Figura 79: Preparación del puesto para la interacción entre el funcionario y el usuario

10.2. Proceso de evaluación en la oficina de la JPT de Toledo

El proceso de evaluación con usuarios reales en la DGT de Toledo necesitó de las siguientes fases:

1. En una primera fase, hay que formar al funcionario en el manejo del programa de traducción de voz a LSE. Además, con el fin de garantizar la mejor tasa de reconocimiento se adaptarán los modelos acústicos a dicho locutor. Para realizar el proceso de adaptación se grabarán 100 frases pronunciadas por el funcionario que va a manejar el sistema. Esta fase requerirá al menos **1 hora** de trabajo con el funcionario que utilizará el sistema.
2. En segundo lugar, hay que explicar a los usuarios el proyecto y la manera de interaccionar con el funcionario. Esta fase requerirá al menos **media hora** de trabajo incluyendo explicación y práctica. Esta segunda fase se puede hacer en paralelo con la primera.
3. En una tercera fase hay que explicar tanto a los usuarios, como al funcionario, el proceso de evaluación. Esta explicación debe incluir:
   a. Descripción de los diferentes escenarios que se van a probar. Se les dará información escrita de dichos escenarios con ejemplos de diálogos traducidos a LSE.
   b. Descripción de los cuestionarios que van a tener que rellenar al final de la evaluación. Se les dará información escrita de dichos cuestionario traducidos a LSE.

   Esta fase necesitará de, al menos, **media hora**.
4. Finalmente se realiza la fase de evaluación en la que un usuario va a realizar la renovación del carné de conducir en los diferentes escenarios planteados. Esta fase estará supervisada, al menos, por un técnico de los programas, un observador externo del proceso de evaluación y un intérprete en LSE. Se estima que se necesite

alrededor de **media hora para completar 6 escenarios por cada uno de los usuarios** que prueben el sistema.

10.3. Escenarios y ejemplos de diálogo para cada escenario

Los diferentes escenarios que se plantean en un proceso de evaluación son situaciones reales que se deben simular con usuarios reales. En nuestro caso, se han planteado 6 escenarios, uno de ellos en los que el proceso de renovación del carné de conducir se realiza sin ninguna incidencia, y otros 5 escenarios en los que aparecen alguna incidencia o problema que impide realizar la solicitud de renovación.

ESCENARIO 1: Renovación del carné sin ninguna incidencia (CARNÉ RENOVAR PROBLEMA NO).

EJEMPLO DE DIÁLOGO:

	Castellano	LSE
Funcionario	Hola, buenos días	HOLA BUENOS DÍAS
Usuario	Hola	HOLA
Usuario	Me gustaría renovar el carné	YO CARNET RENOVAR QUERER
Funcionario	Rellene este impreso, por favor	POR-FAVOR PAPEL ESTE TU ESCRIBIR
Funcionario	Déme la documentación	DOCUMENTACIÓN DAR-A_MI
Funcionario	Tiene que abonar dieciocho euros con cincuenta	PRECIO DIECIOCHO COMA CINCUENTA EUROS
Usuario	¿Tengo que hacer algo más?	ALGO MÁS?
Funcionario	Tome este documento que será su carné provisional	PAPEL ESTE CARNÉ PROVISIONAL
Funcionario	Antes de tres meses le llegará el nuevo carné por correo a casa	CARNET TU CASA ENVIAR-A_TI MÁS-O-MENOS TRES-MESES
Funcionario	Muy bien, ya está todo	TODO BIEN
Usuario	Muchas gracias	MUCHAS-GRACIAS

ESCENARIO 2: Falta la foto (FOTO FALTAR).

EJEMPLO DE DIÁLOGO:

	Castellano	LSE
Funcionario	Hola, buenos días	HOLA BUENOS DÍAS
Funcionario	¿Qué desea?	QUERER QUÉ?
Usuario	Me gustaría renovar el carné	YO CARNET RENOVAR QUERER
Funcionario	Rellene este impreso	PAPEL ESTE TU ESCRIBIR
Funcionario	Déme la documentación	DOCUMENTACIÓN DAR-A_MI
Funcionario	Necesito una fotografía actual	FOTO ACTUAL YO NECESITAR
Usuario	Lo siento no la tengo	LO-SIENTO FOTO HABER-NO
Usuario	¿Dónde hay un fotomatón?	YO FOTOS FLASH DÓNDE?
Funcionario	Vaya allí al fotomatón para sacarse la foto rápidamente y vuelve	TU FOTO FLASH IR FIN VOLVER RÁPIDO
Usuario	Muchas gracias	MUCHAS-GRACIAS

ESCENARIO 3: Faltan datos en el impreso (DATOS PAPEL FALTAR).

EJEMPLO DE DIÁLOGO:

	Castellano	LSE
Funcionario	Hola, buenos días	HOLA BUENOS DÍAS
Funcionario	¿Qué desea?	QUERER QUÉ?
Usuario	Me gustaría renovar el carné	YO CARNET RENOVAR QUERER
Funcionario	Rellene este impreso	PAPEL ESTE TU ESCRIBIR
Funcionario	Falta la fecha y la firma	FECHA FIRMA TU ESCRIBIR FALTA
Funcionario	Déme la documentación	DOCUMENTACIÓN DAR-A_MI
Funcionario	Tiene que abonar dieciocho euros con cincuenta	PRECIO DIECIOCHO COMA CINCUENTA EUROS
Usuario	¿Tengo que hacer algo más?	ALGO MÁS?
Funcionario	No, todo está bien	NO TODO BIEN
Funcionario	Tome este documento que será su carné provisional	PAPEL ESTE CARNÉ PROVISIONAL
Funcionario	Antes de tres meses le llegará el nuevo carné por correo a casa	CARNET TU CASA ENVIAR-A_TI MÁS-O-MENOS TRES-MESES
Funcionario	Muy bien, ya está todo	TODO BIEN
Usuario	Muchas gracias	MUCHAS-GRACIAS
Usuario	Hasta luego	HASTA-LUEGO

ESCENARIO 4: Falta el certificado médico (PAPEL ESPECIAL MÉDICO FALTA).

EJEMPLO DE DIÁLOGO:

	Castellano	LSE
Funcionario	Hola, buenos días	HOLA BUENOS DÍAS
Funcionario	¿Qué desea?	QUERER QUÉ?
Usuario	Me gustaría renovar el carné	YO CARNET RENOVAR QUERER
Funcionario	Rellene este impreso	PAPEL ESTE TU ESCRIBIR
Funcionario	Déme la documentación	DOCUMENTACIÓN DAR-A_MI
Funcionario	Necesito el certificado médico	PAPEL ESPECIAL MÉDICO FALTA
Usuario	Lo siento no lo tengo	LO-SIENTO PAPEL HABER-NO
Usuario	¿Dónde puedo conseguirlo?	PAPEL ESPECIAL MÉDICO DÓNDE?
Funcionario	Hay un centro en la calle Velazquez número 7	CALLE VELAZQUEZ NÚMERO SIETE
Usuario	Me puede repetir	REPETIR
Funcionario	Hay un centro en la calle Velazquez número 7	CALLE VELAZQUEZ NÚMERO SIETE
Usuario	Muchas gracias	MUCHAS-GRACIAS

ESCENARIO 5: Falta el DNI (DNI FALTA).

EJEMPLO DE DIÁLOGO:

	Castellano	LSE
Funcionario	Hola, buenos días	HOLA BUENOS DÍAS
Funcionario	¿Qué desea?	QUERER QUÉ?
Usuario	Me gustaría renovar el carné	YO CARNET RENOVAR QUERER
Funcionario	Rellene este impreso	PAPEL ESTE TU ESCRIBIR
Funcionario	Déme la documentación	DOCUMENTACIÓN DAR-A_MI
Funcionario	Necesito su deneí	TU DNI YO NECESITAR
Usuario	Me puede repetir	REPETIR
Funcionario	Necesito su deneí, es obligatorio	TU DNI NECESITAR OBLIGATORIO
Funcionario	Va a tener que volver usted otro día	LO-SIENTO TU OTRO-DÍA VENIR

Usuario	Muchas gracias	MUCHAS-GRACIAS

ESCENARIO 6: Pagar con tarjeta de crédito (TARJETA_CREDITO PAGAR).

EJEMPLO DE DIÁLOGO:

	Castellano	LSE
Funcionario	Buenas	HOLA BUENOS DÍAS
Funcionario	Dígame	QUERER QUÉ?
Usuario	Me gustaría renovar el carné	YO CARNET RENOVAR QUERER
Funcionario	Rellene este impreso	PAPEL ESTE TU ESCRIBIR
Usuario	Gracias	GRACIAS
Funcionario	Déme la documentación	DOCUMENTACIÓN DAR-A_MI
Funcionario	Tiene que abonar dieciocho euros con cincuenta	PRECIO DIECIOCHO COMA CINCUENTA EUROS
Usuario	¿Puedo pagar con tarjeta de crédito?	TARJETA_CRÉDITO PODER?
Funcionario	No, tiene que ser en efectivo	NO DINERO DEBER
Funcionario	Vaya allí al cajero para sacar dinero rápidamente y vuelva	TU CAJERO IR DINERO FIN VOLVER RÁPIDO
Usuario	Muchas gracias	MUCHAS-GRACIAS

10.4. Evaluación del sistema de traducción de voz a LSE

En este epígrafe se comentará la anotación de eventos, los cambios a realizar en el software, las medidas objetivas y las medidas subjetivas a recoger tanto del funcionario como del usuario.

10.4.1. Anotación de eventos

Se anota cada evento en una línea. Al comienzo de cada línea se escribe la fecha y la hora, y a continuación el botón o el mensaje que se quiere apuntar. Toda la información debe ir entre corchetes. A continuación veremos un fragmento del fichero de depuración:

```
01/16/09 16:07:24: [ESCENARIO 01]
01/16/09 16:07:38: [BOTÓN RECONOCER]
01/16/09 16:07:39: [RECONOCIENDO…]
01/16/09 16:07:39: [TRADUCIENDO…]
01/16/09 16:07:39: [SINTETIZANDO…]
01/16/09 16:07:41: [SIGNANDO…]
01/16/09 16:07:41: [FICHERO:LSE_1_1.wav|hola buenos días (0.95) {934ms}|HOLA BUENOS DÍAS (0.95) {233ms}|EJEMPLOS|0.25|{1478ms}]
…
```

Al final de cada turno anotamos el fichero donde se ha grabado la voz, la salida del reconocedor (incluyendo la medida de confianza y el tiempo en milisegundos que ha tardado en reconocer), y la salida del sistema de traducción junto con la medida de confianza y el tiempo en milisegundos que ha tardado en traducir. Seguidamente se incluye el tipo de traductor utilizado (EJEMPLOS, REGLAS, ESTADÍSTICO), la distancia o la tasa según el modo de traducción, y finalmente, el tiempo que ha tardado en signar (en milisegundos).

10.4.2. Cambios a realizar en el software

Los principales cambios realizados en el software son los siguientes:

- En el interfaz hemos incluido una lista para seleccionar el escenario que se quiera grabar y un botón que permita volcar esta información al fichero de depuración.
- Los eventos a apuntar son los siguientes:
    - Cuando damos en el botón de ESCENARIO.
    - Cuando damos a los botones de RECONOCER, TRADUCIR y ESTIMAR RUIDO.
    - Cuando se envía un mensaje a la pantalla para actualizar el estado del sistema (reconociendo…, traduciendo…, sintetizando…, signando…)
    - Al final imprimimos el fichero donde se graba la voz, la salida del reconocedor, la salida del traductor (con los tiempos en milisegundos y las medidas de confianza), el tipo de traducción utilizada (EJEMPLOS, REGLAS o ESTADÍSTICA), la distancia o tasa, y el tiempo de signado.

10.4.3. Medidas objetivas

Las medidas objetivas calculadas a partir de la información recogida en el fichero de depuración son las siguientes:

- Tasa de reconocimiento: porcentaje de palabras reconocidas correctamente. Esta medida requiere la trascripción de los ficheros de audio.
- Medida de confianza del sistema de reconocimiento de voz.
- Tasa de traducción: porcentaje de glosas generadas correctamente requiere la traducción manual de las frases.
- Medida de confianza del sistema de traducción de voz.
- Tiempo en milisegundos desde que se comienza a reconocer hasta que se obtiene la traducción.
- Tiempo desde que se obtiene la traducción hasta que se termina de signar.
- Tipo de traducción (porcentaje de cada tipo).
- Número de veces que se pulsan los botones de RECONOCER, TRADUCIR,…
- Número de veces que se pulsa el botón TRADUCIR para repetir la última intervención.
- Número de turnos del funcionario para cada uno de los escenarios (número de veces que pulsa el botón RECONOCER o TRADUCIR)

10.4.4. Medidas subjetivas: preguntas en los cuestionarios

Las medidas sujetivas se obtuvieron de preguntas que se rrealizaron, tanto a los usuarios como al funcionario, en un cuestionario que rellenó al final de la prueba cada uno de ellos.

En relación con la traducción de voz a LSE, al funcionario se le han preguntado los siguientes aspectos:

- Rapidez del sistema.
- Tasa de reconocimiento percibida.
- Facilidad de uso.
- Aprendizaje de manejo del sistema.
- Uso futuro en situaciones de necesidad.
- Valoración global del sistema.

A los usuarios se les ha preguntado sobre:

- La inteligibilidad de las frases en LSE.
- La naturalidad en la representación de los signos.
- Tasa de traducción percibida.
- Uso futuro en situaciones de necesidad.
- Valoración global del sistema.

10.4.5. Cuestionarios finales

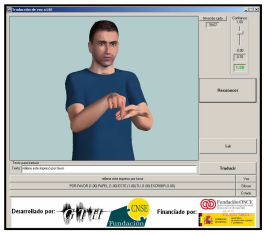
A continuación se muestran los cuestionarios utilizados tanto para el funcionario como para los usuarios.

CUESTIONARIO PARA EL FUNCIONARIO

Datos:

- Edad:_____
- Sexo: Varón ___ Mujer___
- Frecuencia de uso del ordenador:
    - Muy baja____ Baja____ Alta____ Muy Alta____

Valore los siguientes aspectos del sistema de traducción de voz a LSE:



MAL REGULAR BIEN

PREGUNTA	0	1	2	3	4	5
Rapidez del sistema						
Tasa de reconocimiento						
Facilidad de uso						
Rapidez de aprendizaje para el manejo del sistema						
Si no hubiera un intérprete de LSE, ¿usaría este sistema?						
Valoración GLOBAL						
Comentarios adicionales:						

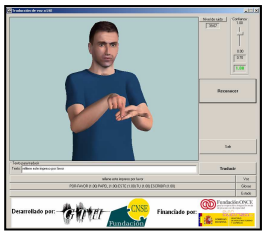
MUCHAS GRACIAS

CUESTIONARIO PARA EL USUARIO

Datos:

- Edad:_____ Conductor: Sí _____ No ___
- Sexo: Varón ___ Mujer___
- Frecuencia de uso del ordenador:
  - Nunca____ Poco____ A veces____ Mucho ____
- Comprensión del castellano escrito:
  - Baja____ Normal____ Alta
- Comprensión de las glosas:
  - Baja____ Normal____ Alta

Valore los siguientes aspectos del sistema de traducción de voz a LSE:



MAL REGULAR BIEN

PREGUNTA	0	1	2	3	4	5
Los signos son correctos						
Los signos se entienden						
Los signos son naturales						
Si no hubiera un intérprete de LSE, ¿usaría este sistema?						
NOTA FINAL						
Comentarios:						

MUCHAS GRACIAS

10.5. Resultados finales

La evaluación se llevó acabo en las instalaciones de la Dirección General de Tráfico (DGT) de Toledo. El sistema fue evaluado por 10 personas sordas que interaccionaron con 2 funcionarios de la DGT a través de nuestro sistema. Estas 10 personas probaron la mayoría de los 6 escenarios planteados con anterioridad, generando 48 diálogos entre un funcionario y una persona sorda. En relación con las principales características de los usuarios cabe comentar los siguientes aspectos:

- La edad de los usuarios cabe comentar que oscilaba entre 22 y 55 años con una media de 40,9 años.
- La mayoría son varones (6 de 10) y están en posesión del carné de conducir (8 de 10).
- En cuanto al uso del ordenador cabe comentar que lo utilizan con bastante frecuencia.
- La mayoría posee una buena comprensión del castellano escrito y de la utilización de glosas para la especificación de los signos.
- La mitad de los usuarios eran de Madrid y la otra mitad de Toledo.

En la siguiente figura podemos ver algunas de las fotos recogidas durante la evaluación:

Figura 80: Fotos de la evaluación en la Jefatura Provincial de Tráfico de Toledo

AGENTE	MEDIDA	VALOR
Sistema	Tasa de reconocimiento	95,2%
	Tasa de traducción	91,1%
	Tiempo de reconocimiento	3,3 seg
	Tiempo de traducción	0,0013 seg
	Tiempo de signado	4,7 seg
	Traducción basada en ejemplos	94,9%
	Traducción basada en reglas	4,2%
	Traducción estadística	0,8%
	Porcentaje de turnos con reconocimiento de voz	92,4%
	Porcentaje de turnos con traducción desde texto	0%
	Porcentaje de turnos de repetición (sin reconocimiento de voz)	7,6%
	Número de turnos del funcionario por diálogo	8,4
	Número de diálogos	48

Tabla 29: Resultados de las medidas objetivas.

En relación con las medidas objetivas, los resultados se muestran en la Tabla 29. Como se puede observar se ha conseguido una tasa de reconocimiento de voz superior al 95% y una tasa de traducción del 91%. Por otro lado el funcionamiento del sistema es muy rápido con tiempos de unos pocos segundos para completar el proceso de traducción de voz a LSE: reconocimiento de la voz, la traducción de las palabras a glosas y la representación posterior de los signos con el agente animado. Estos números aseguran una calidad apropiada para poder llegar a realizar estas pruebas reales.

En cuanto a las estrategias de traducción se puede observar que en este caso la traducción basada en ejemplos ha sido la predominante lo que refleja el buen estudio lingüístico realizar en las primeras tareas del proyecto. Este estudio permitió recoger, con un alto grado de fiabilidad, las expresiones que con mayor frecuencia pronuncia el funcionario en este tipo de diálogos.

En cuanto a la tipología de turnos podemos observar que en el 7,6% de los turnos el funcionario no tuvo que volver a pronunciar la frase, fue suficiente con pulsar el botón de traducir el texto anterior y el sistema vuelve a generar los mismos signos. El porcentaje de turnos con traducción de texto es cero lo que indica que no fue necesario que el funcionario introdujera mediante el teclado la frase a traducir. En todos los casos el sistema reconoció la frase con el suficiente grado de fiabilidad como para ser traducida con pocos errores (se entendió).

En relación con las medidas subjetivas los resultados son:

AGENTE	MEDIDA	VALOR (0-5)
Funcionario	Rapidez del sistema	4
	Tasa de reconocimiento	3,5
	Facilidad de uso	3,5
	Rapidez de aprendizaje para el manejo del sistema	3,5
	Si no hubiera un intérprete de LSE, ¿usaría este sistema?	3,5
	Valoración GLOBAL	3,5
Usuario	¿Los signos son correctos?	2,1
	Comprendo los signos	2,2
	El signado es natural	0,8
	Si no hubiera intérprete, ¿usaría el sistema?	2,0
	Valoración GLOBAL	2,2

Tabla 30: Resultados de medidas subjetivas.

La valoración del funcionario es bastante positiva ofreciendo en todos los aspectos una puntuación de 3,5. Quizás el comentario más importante es que a veces era incómodo tener la pantalla mirando hacia el usuario (ver Figura 81). Es cierto que el sistema va realimentando con voz la frase reconocida y emite un pitido cuando termina de signar, para que el funcionario continúe interviniendo, pero quizás en el futuro haya que plantear conectar una doble pantalla para permitir al funcionario observar con detalle lo que va ocurriendo.

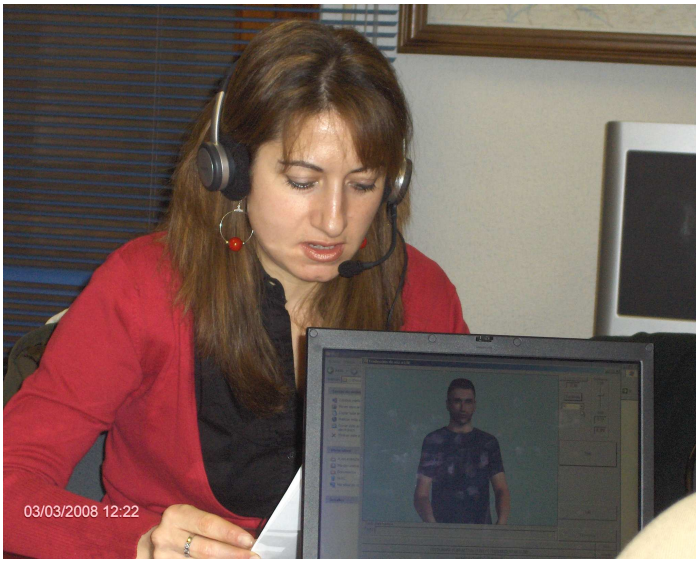


Figura 81: Intervención de Marta (la funcionaria de la JPT de Toledo) con la pantalla del Tablet PC girada hacia el usuario.

En relación con la valoración de los usuarios podemos comentar que es bastante baja, rozando el aprobado en la valoración global. Quizás la crítica más importante recae sobre la naturalidad en la representación de los signos (con una puntuación de 0,8). En relación con este aspecto hay bastante unanimidad: el sistema requiere la inversión de un mayor esfuerzo en el diseño del agente animado para incrementar su naturalidad y expresividad.

En relación con la inteligibilidad de los signos se consigue una mayor nota (2,1 y 2,2) pero aun así, sigue siendo baja. Los principales motivos de estas valoraciones son los siguientes:

- Sin duda la poca naturalidad / expresividad del agente animado hace que la comprensión de los signos sea menor. Para mejorar esta naturalidad es necesario seguir invirtiendo esfuerzo en mejorar la flexibilidad y movimientos del agente animado, sobre todo en la parte de la cara.
- Por otro lado, también es justo reconocer que con bastante frecuencia se produjeron discrepancias entre los propios usuarios sobre la correcta ejecución de algunos de los signos (por ejemplo, el signo FOTO, sobre la flexión del dedo índice de ambas manos o sólo de la mano derecha) o en la utilización de uno u otro signo (por ejemplo la utilización del signos FECHA en lugar de DÍA). Estas diferencias se resuelven en la práctica remarcando la expresividad o la pronunciación en el signo, características que nuestro agente animado tiene que mejorar sensiblemente. El signado realizado por el sistema se ha apoyado en el diccionario DILSE III generado por la Fundación CNSE. Estas discrepancias deben animar a seguir trabajando en la estandarización de la Lengua de Signos Española (LSE). Aunque no existen datos relevantes significativamente, sí que se percibió un mayor grado de acuerdo entre las personas procedentes de Madrid.

- Otra fuente de discrepancias entre los usuarios fue la estructura de algunas frases. La LSE, al igual que cualquier otra lengua, ofrece una importante flexibilidad. Dicha flexibilidad a veces no es entendida de esta manera y algunas de las posibilidades se pueden considerar como erróneas en lugar de frases alternativas. Algunos ejemplos son:
    - En el caso de la pregunta "¿Qué desea?" que se puede traducir a LSE como QUERER QUÉ? o como TU QUERER?. El sistema utilizaba la primera de ellas aunque para algunos usuarios les parecía raro y preferían la segunda.
    - La utilización del signo CAJERO y la necesidad o no de ir acompañado por el signo BANCO o DINERO para que se entienda completamente.
    - La utilización de los signos FOTO FLASH para hacer referencia a un fotomatón en lugar de CABINA.
    - En la frase "DNI CARNET CONDUCIR LOS-DOS DAR-A_MI" también se produjo problemas al entender el signo LOS-DOS a qué hacía referencia. Se sugirió cambiarlo de posición o incorporar algún signo explicativo adicional.
- El agente animado presenta excesiva rigidez en la ejecución de algunos de los signos que hace que el ángulo de presentación del agente animado pueda influir en la percepción de algunos signos. Por ejemplo, en el caso del signo VENIR (en el que hay un movimiento que combina un desplazamiento hacia abajo y otro hacia el signante), si la orientación del avatar es perfectamente enfrentada con el usuario, los movimientos de profundidad no se perciben, observándose únicamente los movimientos verticales. Para resolver este problema se optó por girar ligeramente el agente animado para percibir todos los movimientos.
- Existe un conjunto de signos (signos deícticos: que hacen referencia a una persona, cosa o lugar situados en una posición concreta) cuya ejecución implica la señalización de un objeto en una dirección concreta, por ejemplo, cuando se hace referencia a "esta ventanilla" traducido por ESTE VENTANILLA. El signo ESTE deberá ejecutarse en la dirección donde esté situada la ventanilla. Esta situación es muy dependiente del entorno donde esté situado el sistema. Para resolver este problema se evitaron este tipo de signos ofreciendo una especificación mayor del objeto concreto: "ventanilla de conductores" en LSE "VENTANILLA ESPECÍFICO CONDUCTOR".

Todos estos motivos que dificultan la percepción de los signos suponen que el reconocimiento de voz y la traducción a glosas se han realizado sin errores. Pero, ¿cómo influye un error de traducción en la comprensión de las frases en LSE?. Pues la respuesta es que la influencia es decisiva. Si ya hay dificultades de comprensión por los motivos comentaros anteriormente, la introducción de un error en la secuencia de glosas hace que el usuario se despiste completamente y no entienda el significado de la frase:

- En el caso de que se produzca la inserción de una glosa errónea o la sustitución de una glosa por otra, la consecuencia es desastrosa. En la mayoría de los casos, el usuario rechaza completamente la frase y solicita la repetición de la misma.
- En el caso del borrado de alguna de las glosas, el resultado no es tan drástico y en algunos casos el usuario ha sido capaz de deducir el significado de la frase en LSE aun faltando algún signo.

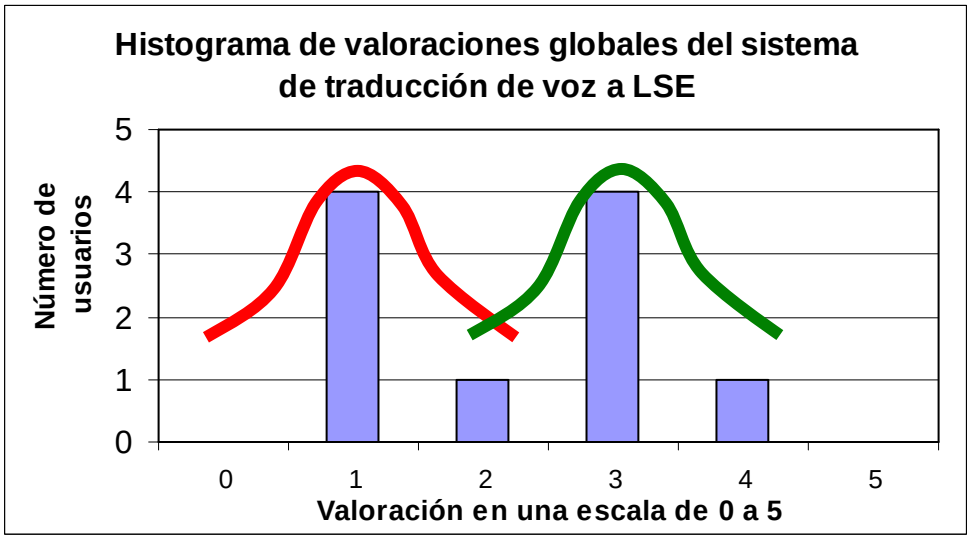


Figura 82: Distribución de número de usuarios según la valoración global del sistema.

Para terminar la discusión sobre los resultados de valoración del sistema de traducción de voz a LSE, en la Figura 82 se representa el número de usuarios en función de la valoración global ofrecida al sistema. Como se puede observar existen dos tipologías de usuarios muy diferenciadas: un grupo de usuarios que valoran positivamente (3,2) el sistema mientras que otro grupo de usuarios lo valoran negativamente (1,2).

En líneas generales se puede decir que aunque las valoraciones reflejan la necesidad de importantes mejoras, los comentarios recibidos sí que reflejan un interés importante por este tipo sistemas, lo que anima a seguir invirtiendo esfuerzo para abordar las mejoras necesarias.

Finalmente, una conclusión muy importante del proyecto es que se debe complementar los sistemas desarrollados (de traducción de voz a LSE) con sistemas que permitan traducir LSE a voz. Estos sistemas permitirían hacer llegar al funcionario las preguntas que desea formular el usuario sin necesidad de utilizar un intérprete.

11. Conclusiones generales del proyecto

En este apartado es importante remarcar que la conclusión más importante del proyecto es que se ha conseguido un grado de cumplimiento de los objetivos propuestos del 100%, llevando acabo todas las tareas propuestas y generando todos los resultados previstos. Los principales resultados del proyecto han sido:

- Una arquitectura software de traducción de voz a LSE.
- Dos demostradores que permiten la traducción de voz a LSE aplicados a traducir las frases de un funcionario en dos servicios públicos de atención personal: el servicio de información y gestión para la renovación del Documento Nacional de Identidad (DNI) y el servicio de renovación del carné de conducir en la Dirección General de Tráfico (DGT).
- Evaluación del sistema de traducción de voz a LSE con usuarios finales que han aportado importantes comentarios sobre los aspectos que son necesarios mejorar para convertir estos demostradores en productos reales.
- Un entorno de edición para la generación de signos con un agente animado. Dicho entorno permite la traducción de un signo (representado en alguno de los estándares de signo-escritura considerados) a los movimientos del agente animado. Este entorno incorpora un sistema de conversión entre los dos estándares de signoescritura: SEA (Sistema de Escritura Alfabética) y HamNoSys (estándar propuesto desde la Universidad de Hamburgo)

Los principales contenidos multimedia generados:

- Un corpus paralelo con las frases en castellano y su traducción en LSE para los dos dominios de aplicación estudiados (renovación del DNI y del carné de conducir). Además se incluyen vídeos grabados para una de las frases con el fin de especificar los signos concretos en LSE.
- Generación de una base de datos con las descripciones, características, significado y animación de los signos correspondientes a los servicios públicos de atención abordados en este proyecto: renovación del DNI y del carné de conducir.

Los documentos generados:

- Informe final del proyecto donde se recogen todos los detalles del proyecto, de las aplicaciones desarrolladas y los contenidos digitales generados.
- Manuales de usuario de todos los programas generados.
- Documento con las normas de etiquetado en glosas de los corpora.
- Documentación para la impartición de cursos del estándar de signo-escritura HamNoSys.

- Descripción de los contenidos digitales generados.

De las pruebas realizadas se desprende que las prestaciones de los sistemas de reconocimiento de voz y de traducción de castellano a LSE en los dos dominios de aplicación considerados han sido muy buenas, tanto en tasa de reconocimiento y traducción, como en velocidad de ejecución. Este hecho confirma que el estado de estas tecnologías es suficiente para ser incorporadas con éxito en este tipo de sistemas.

La calidad de agente animado debe ser sensiblemente mejorada para garantizar la correcta comprensión de los signos por los usuarios. Esta mejorada debe ir enfocada a incrementar la naturalidad de los movimientos y a incorporar una mayor expresividad del cuerpo y de la cara.

Para la automatización del proceso de traducción entre lenguas es muy importante apoyarse en estándares suficientemente aceptados y extendidos entre todas las personas que usan esa lengua. Desde más de 20 años la Fundación CNSE ha invertido un gran esfuerzo en la normalización de la LSE: signos y gramática. Esfuerzo que debe continuar. Además se debe extender este esfuerzo de normalización a la definición de un estándar de signo escritura que tenga una gran aceptación entre los usuarios.

Finalmente, una conclusión muy importante del proyecto es que se debe complementar los sistemas desarrollados (de traducción de voz a LSE) con sistemas que permitan traducir LSE a voz. Estos sistemas permitirían hacer llegar al funcionario las preguntas que desea formular el usuario sin necesidad de utilizar un intérprete.

En la actualidad se continúa realizando la labor de divulgación de los resultados del proyecto. Esta labor se está concretando en los siguientes aspectos:

- Redacción y envío de artículos científicos a las diferentes revistas de investigación.
- Redacción de notas de prensa para enviar a los diferentes medios de comunicación.
- Actualización de la página web del proyecto (www.traduccionvozlse.es).
- Preparación de actos donde se presenten los principales resultados del proyecto y a los que se invitará a varios medios de comunicación.

12. ANEXO I: Manual de usuario del sistema de traducción de voz a LSE

12.1. Requisitos del programa

Para poder instalar el programa de traducción de voz en castellano a LSE, se necesitarán unos requisitos mínimos del sistema:

- **Procesador:** Pentium II o superior.
- **Sistema Operativo:** Windows 98/2000/Xp.
- **Memoria:** 512 MB de RAM.
- **Espacio en disco:** 100 MB para la instalación de la aplicación, además de 150 MB para la instalación de programas adicionales.
- **Unidades de disco:** unidad de CD-ROM.
- **Pantalla:** resolución de 800 x 600 píxeles o superior.

Además de estos requisitos hardware del sistema, serán necesarias ciertas aplicaciones y utilidades.

Para poder visualizar correctamente las fuentes de la aplicación, habrá que instalar dos fuentes de letras, una para los símbolos de HamNoSys y otra para el conjunto de caracteres de SEA (Sistema de Escritura Alfabética). En el apartado siguiente explicaremos sus pasos de instalación.

Para poder representar los diferentes signos mediante el agente animado, habrá que instalar el paquete del control ActiveX SiGMLSigning. Los pasos de instalación de este paquete los describiremos en el apartado dedicado a la instalación.

Será necesario tener instalada en el sistema una Máquina Virtual Java, en su versión 1.4 o superior. Su descarga es gratuita, accediendo desde el siguiente link: *http://www.java.com/es/download/*.

Finalmente, para realizar la conversión de texto a voz, es necesario tener instalado:

- La Microsoft Speech API 5.1 o superior de Microsoft . Su descarga es gratuita accediendo a la web www.microsoft.com.
- La voz de *jorge* del conversor de texto a voz de la empresa Loquendo.

12.2. Instalación de programas

Primero comenzaremos describiendo los pasos a seguir para instalar el programa de traducción de voz a LSE, para finalizar comentando cómo instalar las utilidades necesarias para su correcto funcionamiento.

12.2.1. Traductor de voz a LSE

Para instalar el traductor tenemos que cargar el instalador, pulsando *Setup.exe*. Este instalador se encuentra en la carpeta raíz, y nos irá indicando todos los pasos a seguir para instalar el traductor en nuestro ordenador. El propio instalador será el encargado de instalar las fuentes necesarias para la correcta visualización del programa.

12.2.2. Agente animado

Para instalar el agente animado, necesario para la representación de signos dentro del programa, debemos lanzar el instalador, que encontraremos dentro de la carpeta *SiGMLSigning* (el primer paso es descomoprimir el fichero *.zip proporcionado). El archivo que debemos ejecutar es *Setup.exe*. Nos irá explicando paso por paso todo lo que debemos realizar para la correcta instalación del agente animado.

12.3. Ventana principal del traductor de voz a LSE

Esta aplicación tiene una ventana principal, desde la que podremos lanzar el reconocimiento de voz y ver el resultado de la traducción. Esta ventana principal tiene el siguiente aspecto:

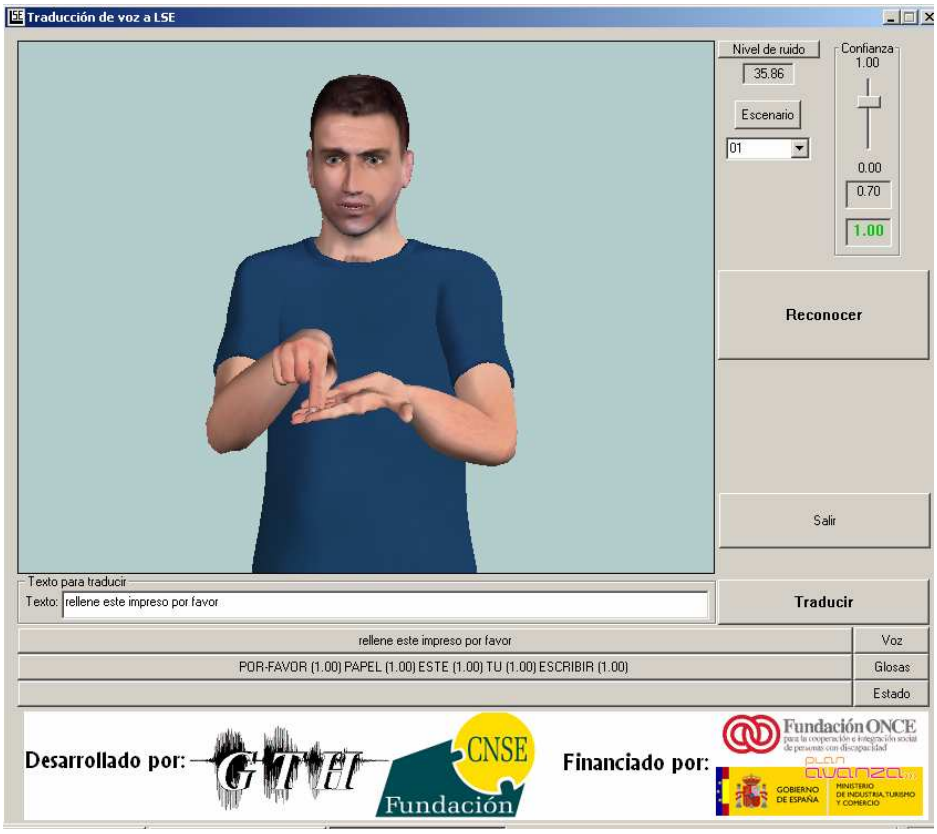


Figura 83: Interfaz principal de la aplicación

En la imagen anterior podemos ver los elementos principales del programa, los botones para lanzar la traducción desde un texto o con reconocimiento de voz, el botón de salir o el de estimación de ruido.

12.3.1. Traducción de la voz pronunciada por el funcionario

Para realizar la traducción de la voz se debe lanzar el reconocimiento de voz, para ello se utiliza el botón “Reconocer”.

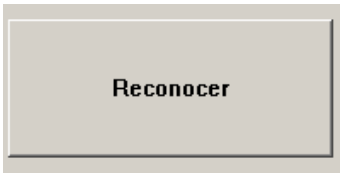


Figura 84: Botón Reconocer

En este momento, se puede comenzar a hablar. El sistema detectará automáticamente el comienzo y final de la voz. Una vez detectado el final se completará el reconocimiento de la frase pronunciada y se procederá a traducir dicha frase a glosas de LSE. Estas glosas re representarán mediante el agente animado. Si desde el teclado del ordenador pulsamos la tecla INTRO, también conseguiremos lanzar el proceso de reconocimiento (es equivalente a pulsar el botón Reconocer).

En las ventanas horizontales que encontramos en la parte de abajo del interfaz podemos ir viendo el resultado del reconocimiento y de la traducción. Las ventanas son las siguientes:

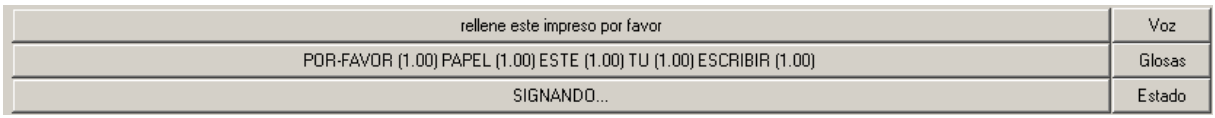


Figura 85: Ventanas horizontales

La primera de ellas va mostrando el resultado del proceso de reconocimiento. La segunda muestra el resultado de la traducción. Este resultado se muestra como una secuencia de glosas con un número entre paréntesis para cada una de las glosas. Este número muestra el nivel de confianza del sistema en la generación de esa glosa como resultado del proceso de traducción. El nivel confianza es un valor entre 0.0 (baja confianza) y 1.0 (alta confianza). Finalmente, la tercera ventana muestra el estado de la aplicación, en este caso SIGNANDO...

La aplicación pasa por 4 estados diferentes:

1. El primero de ellos es el de RECONOCIENDO.... El sistema ha detectado que la persona está hablando y comienza a reconocer.

2. Cuando termina de reconocer el sistema realimenta al funcionario pronunciado la frase reconocida mediante conversión de texto a voz: SINTETIZANDO El TEXTO.

3. De forma paralela a la conversión de texto a voz se realiza la traducción a glosas: TRADUCIENDO...

4. Por último, el resultado de la traducción se representa mediante el agente animado. En este caso el estado es SIGNANDO...

Una vez finalizados los 4 pasos el sistema emite un sonido (ding.wav) para advertir al funcionario que ha terminado de traducir y representar el resultado con el agente animado. Este sonido puede ser de utilidad si la pantalla del ordenador está girada para que el usuario sordo la pueda ver.

Durante el proceso de reconocimiento de voz es posible parar el sistema sin más que pulsar el botón de parar:

Figura 86: Botón de parar

12.3.2. Estimación del nivel de ruido

Para el correcto funcionamiento del sistema de reconocimiento de voz es indispensable hacer una estimación del nivel de ruido de la sala donde está funcionando el sistema. Para realizar esta estimación se puede pulsar el botón de “Nivel de Ruido”

Figura 87: Estimación del ruido

El sistema, al arrancar, realiza una primera estimación del nivel de ruido. Si consideramos que dicha estimación no se ha realizado correctamente o bien pensamos que el ruido de la sala ha cambiado, podemos pulsar el botón para que se repita dicha estimación. Es MUY IMPORTANTE remarcar que durante los 2-3 segundos en los que el sistema está re-estimando el nivel de ruido (tras pulsa el botón de Nivel de ruido) el micrófono debe estar funcionando y encendido pero sin que nadie hable por él, de forma que sólo se esté grabando el ruido de fondo de la sala.

El nivel de ruido depende del micrófono utilizado y de la configuración de los niveles de entrada en la tarjeta de audio del ordenador. En cualquier caso valores inferiores a 30 dB indican que el micrófono no se ha encendido o no está funcionando y valores por encima de 50dB indican que alguien estaba hablando en el momento de la estimación de ruido.

12.3.3. Nivel de confianza

Es posible establecer un filtro según el nivel de confianza medio de la secuencia de glosas generadas. Para ello se utilizan los siguientes controles de la interfaz.

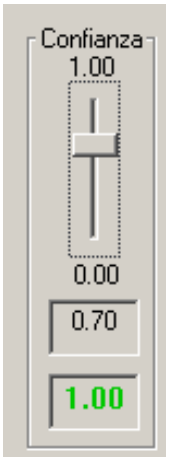


Figura 88: Controles del nivel de confianza.

En la ventana inferior, en verde, se muestra la confianza media de las glosas obtenidas como resultado del proceso de traducción. Mientras que en la ventana inmediatamente superior se muestra el valor límite considerado (0.70 en nuestro caso). Este valor se puede modificar con la barra de desplazamiento vertical. De esta forma, aquellas frases que no obtengan un mínimo nivel de confianza (en media de sus glosas) no serán representadas por el agente animado. El valor mínimo del nivel de confianza se puede modificar desde la interfaz con la barra de desplazamiento.

12.3.4. Traducción de texto a LSE

Además de traducir la voz del funcionario se puede traducir un texto escrito en una ventana de edición proporcionada en la interfaz.

Figura 89: Ventana de edición y botón de traducir el texto de la ventana

Con esta utilidad el funcionario puede traducir a LSE cualquier frase escrita en la ventana de edición. Esta utilidad es muy interesante si en algún momento el sistema no es capaz de reconocer una frase concreta del funcionario. Un caso concreto puede ocurrir cuando el funcionario quiere informar a la persona sorda sobre el nombre de una calle o de un sitio que no está entre las posibilidades de reconocimiento consideradas en la aplicación.

Un aspecto importante que conviene comentar es que cuando se lanza el reconocimiento de voz mediante el botón “reconocer” (comentado anteriormente), el resultado del reconocimiento se copia a esta ventana de edición. Con esta funcionalidad queremos facilitar la posibilidad de que el usuario sordo solicite al funcionario la repetición del último comando. En este caso no será necesario que el funcionario vuelva a pronunciar la frase sino que

bastará con que pulse el botón de traducir para que se vuelva a traducir a LSE la frase pronunciada anteriormente.

12.3.5. Botón salir

El botón salir permite salir de la aplicación.

Figura 90: Botón salir

12.3.6. Selección del escenario

Por último comentar un aspecto que se añadió a la interfaz para el proceso de evaluación.

Figura 91: Controles para la selección del escenario

El proceso de evaluación se realizó considerando varios escenarios de uso y probando el sistema en dichos escenarios. Un escenario no es más que una situación real en la que nos podemos encontrar cuando un usuario sordo quiere renovar el carné de conducir. Para la evaluación se han definido hasta 6 escenarios. Pues bien estos controles permiten seleccionar el escenario que se está simulando en la evaluación con el fin de saber el comportamiento del sistema en cada uno de los escenarios.

12.4. Principales problemas

En la tabla siguiente podemos ver los principales problemas que pueden aparecer en la aplicación, así como sus causas y su posible solución.

PROBLEMAS	CAUSAS	SOLUCIÓN
El agente animado no aparece	No se ha instalado el control ActiveX VGuido o la Máquina Virtual Java	Instalar los dos elementos. Mirar apartado de instalación
Los símbolos de HamNoSys o de SEA	Las fuentes no están	Instalar en FONTS los archivos *hamnosys4.ttf* y

no apareen	instaladas	*sea.ttf*
No se oye la voz en la conversión de texto a voz	No está instalada la voz de jorge de Loquendo o no está instalada la Speech API 5.1 de Microsoft. También es posible que no esté conectados o encendidos los altavoces	Instalar los programas necesario Conectar y encender los altavoces

Tabla 31: Resolución de problemas

13. ANEXO II: Manual de usuario del Editor de Signos

13.1. Requisitos del programa

Para poder instalar el programa *Editor SEA-HamNoSys*, se necesitarán unos requisitos mínimos del sistema:

- **Procesador:** Pentium II o superior.
- **Sistema Operativo:** Windows 98/2000/Xp.
- **Memoria:** 512 MB de RAM.
- **Espacio en disco:** 200 MB para la instalación de la aplicación, además de 150 MB para la instalación de programas adicionales.
- **Unidades de disco:** unidad de CD-ROM.
- **Pantalla:** resolución de 800 x 600 píxeles o superior.

Además de estos requisitos hardware del sistema, serán necesarias ciertas aplicaciones y utilidades.

Para poder visualizar correctamente las fuentes de la aplicación, habrá que instalar dos fuentes de letras, una para los símbolos de HamNoSys y otra para el conjunto de caracteres de SEA (Sistema de Escritura Alfabética). En el apartado siguiente explicaremos sus pasos de instalación.

Para poder representar los diferentes signos mediante el agente animado, habrá que instalar el paquete del control ActiveX SiGMLSigning. Los pasos de instalación de este paquete los describiremos en el apartado dedicado a la instalación.

Será necesario tener instalada en el sistema una Máquina Virtual Java, en su versión 1.4 o superior. Su descarga es gratuita, accediendo desde el siguiente link: *http://www.java.com/es/download/*.

Los gestos de la cara y del cuerpo podrán ser mostrados en vídeos, para lo que será necesario el reproductor QuickTime, programa gratuito que podemos descargarnos en el siguiente link: *http://www.apple.com/quicktime/download/*.

Finalmente, es necesario tener instalado el Microsoft Word para poder visualizar un fichero de ayuda.doc con fotos de ciertos gestos con las cara.

13.2. Instalación de programas

Primero comenzaremos describiendo los pasos a seguir para instalar el programa *Editor SEA-HamNoSys*, para finalizar comentando cómo instalar las utilidades necesarias para su correcto funcionamiento.

13.2.1. Editor SEA-HamNoSys

Para instalar el editor tenemos que cargar el instalador, pulsando *Setup.exe*. Este instalador se encuentra en la carpeta raíz, y nos irá indicando todos los pasos a seguir para instalar el editor en nuestro ordenador. El propio instalador será el encargado de instalar las fuentes necesarias para la correcta visualización del programa.

13.2.2. QuickTime

Dentro de la carpeta *QuickTime*, encontraremos el instalador, *Setup.exe*, el cual nos guiará por todos los pasos necesarios. Este programa es necesario para poder reproducir los videos de muestra de los gestos de expresividad de cara que podremos añadir al agente animado.

13.2.3. Agente animado

Para instalar el agente animado, necesario para la representación de signos dentro del programa, debemos lanzar el instalador, que encontraremos dentro de la carpeta *SiGMLSigning*. El archivo que debemos ejecutar es *Setup.exe*. Nos irá explicando paso por paso todo lo que debemos realizar para la correcta instalación del agente animado.

13.3. Ventana principal del Editor de Signos

Esta aplicación tiene una ventana principal, desde la que podremos trabajar en la generación de signos, además de acceder a las demás utilidades. Esta ventana principal tiene el siguiente aspecto:

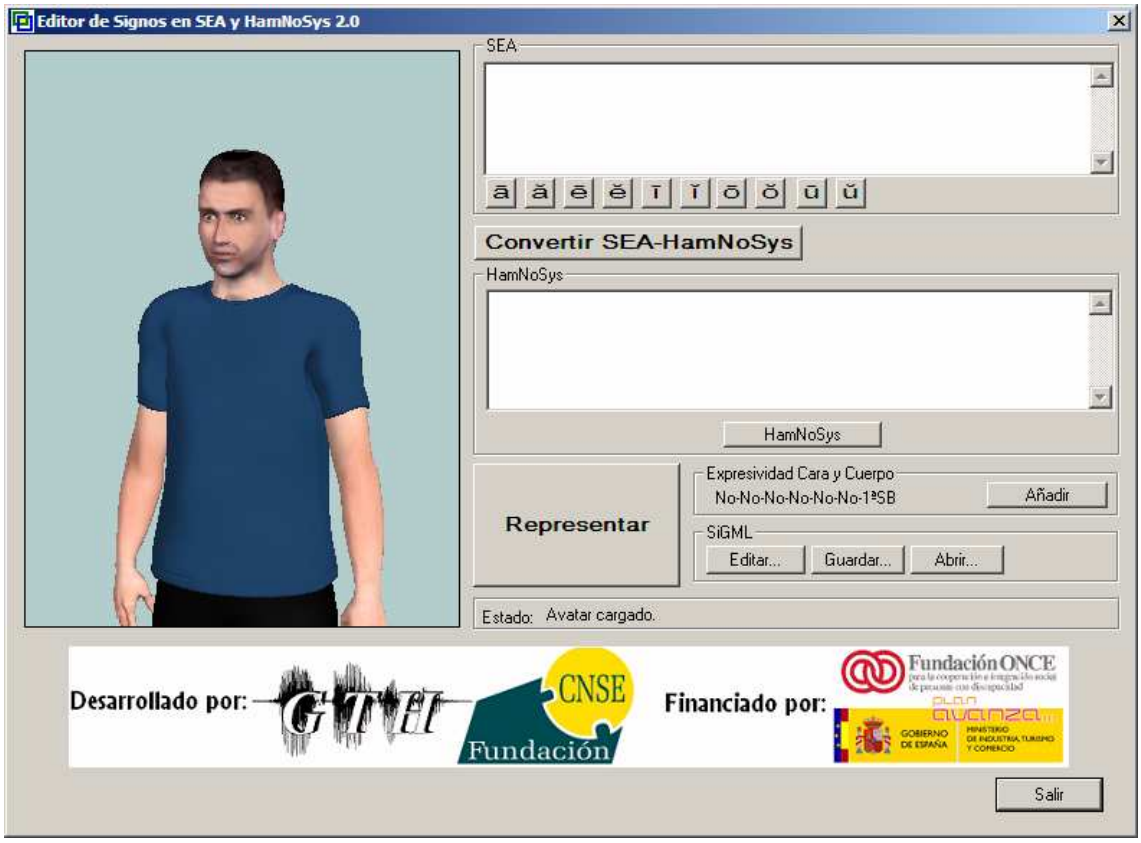


Figura 92: Interfaz principal de la aplicación

En la imagen anterior podemos ver los elementos principales del programa, los dos cuadros de texto para editar la descripción del signo mediante distintos tipos de signo-escritura, y el agente animado que representará los signos. Los signos, en su representación escrita, podrán ser insertados en nuestro programa tanto en HamNoSys como en SEA. Para cada una de estas dos representaciones tenemos su cuadro de texto correspondiente. Hay que tener en cuenta que el agente animado únicamente acepta una representación de signo-escritura en HamNoSys, por lo que siempre que queramos visualizar el signo que estemos editando, será necesario haber introducido en el cuadro de texto de HamNoSys su representación. Si especificamos un signo en SEA, deberá ser previamente convertido a HamNoSys Debido a este hecho, comenzaremos explicando el modo de edición en HamNoSys.

13.3.1. Modo de edición en HamNoSys

Con este modo nos referimos a la edición completa de un signo en su representación en HamNoSys, obviando también su posible traducción desde el SEA, que comentaremos más adelante.

La representación de un signo en HamNoSys se compone de una sucesión de ciertos símbolos, tales como éstos:

Estos caracteres no aparecen en el teclado corriente, por lo que será necesario desplegar una ventana secundaria en la que poder escogerlos.

Para desplegar esta ventana, debemos apretar el botón en el que está escrito *HamNoSys*, que se encuentra justo debajo del cuadro de texto de HamNoSys.

Figura 93: Botón HamNoSys

Una vez apretado este botón, se desplegará una ventana secundaria en la que aparecerá el teclado de HamNoSys.

Figura 94: Teclado de HamNoSys

Esta ventana se compone de seis pestañas que agrupan todos los símbolos de HamNoSys (Forma, Orientación, Localización, Movimiento 1, Movimiento 2, Mano Pasiva) y de dos botones (Espacio, Retroceso) comunes a todas ellas.

Dentro de cada una de estas pestañas aparecen los símbolos de HamNoSys correspondientes, con los que poder especificar un signo completo.

- En la pestaña *Forma* se encuentran los símbolos de HamNoSys con los que especificar la forma de la mano.

Figura 95: Pestaña *Forma* del teclado de HamNoSys

- En la pestaña *Orientación* encontramos los caracteres de HamNoSys con los que podemos especificar la orientación a la que apunta el eje de la mano, y dentro de ese eje, hacia dónde mira la palma.

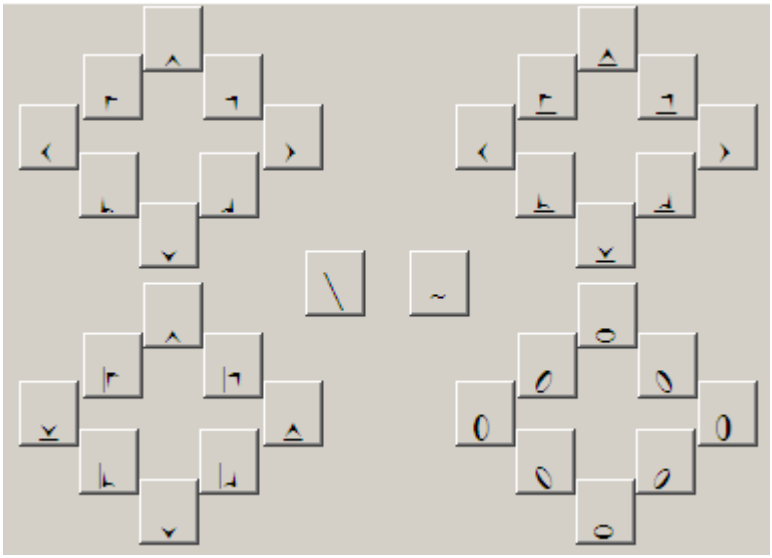


Figura 96: Pestaña *Orientación* del teclado de HamNoSys

- En la pestaña *Localización* encontramos todos los símbolos con los que poder especificar dónde se coloca la mano, pudiendo a su vez detallar los contactos, tanto con el cuerpo como entre las dos manos.

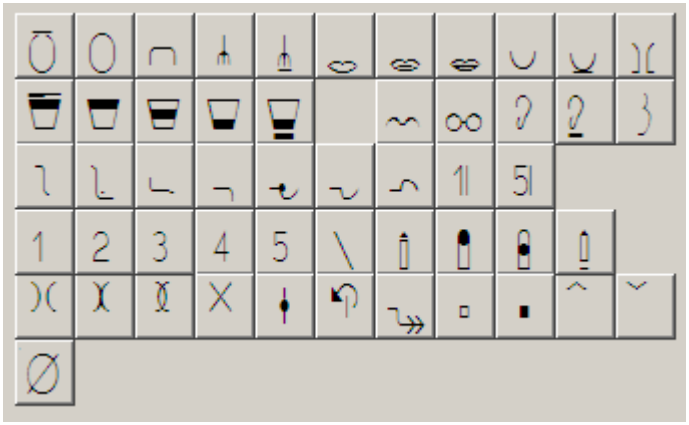


Figura 97: Pestaña *Localización* del teclado de HamNoSys

- En la pestaña *Movimiento 1* podemos encontrar los principales movimientos rectilíneos que puede realizar la mano, así como varios modificadores, tales como variaciones de velocidad, realizar curvaturas en los movimientos rectos, o introducir repeticiones.

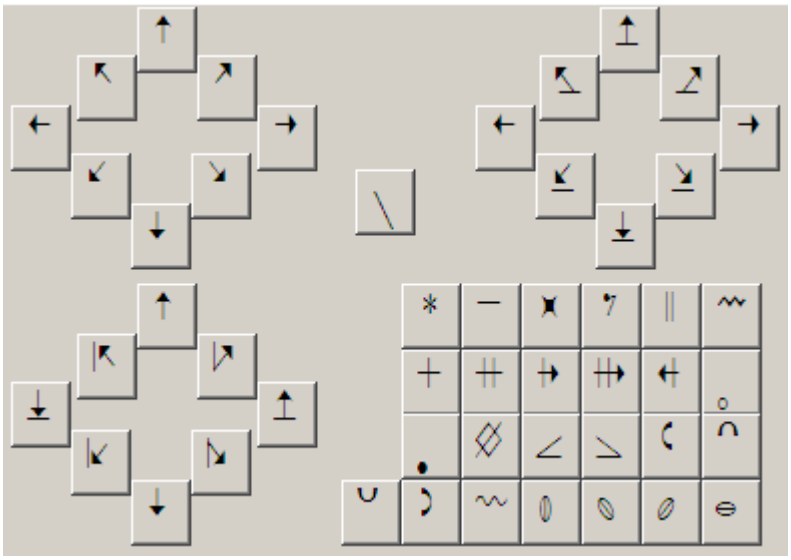


Figura 98: Pestaña *Movimiento 1* del teclado de HamNoSys

- En la pestaña *Movimiento 2* se encuentran todos los movimientos curvilíneos que se pueden realizar con la mano, además de varios movimientos de diapasón, abatimiento o juego de dedos.

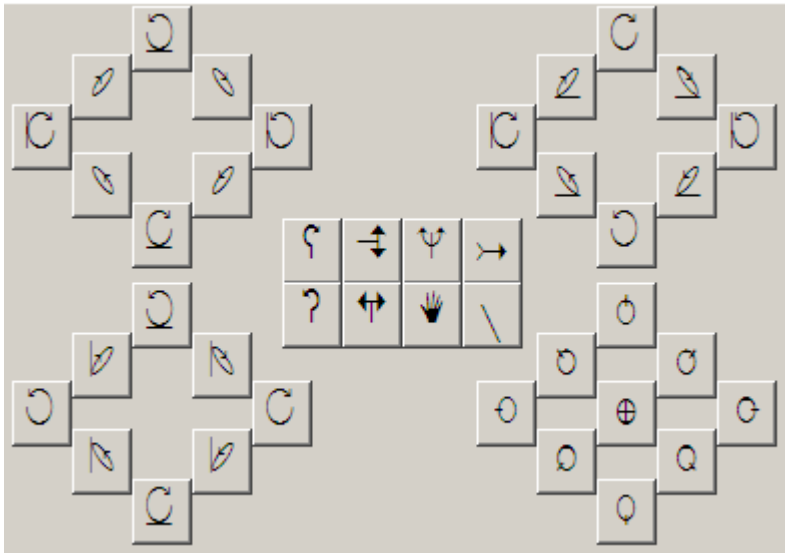


Figura 99: Pestaña *Movimiento 2* del teclado de HamNoSys

- En la sexta pestaña, *Mano Pasiva*, encontramos los elementos de simetría necesarios para especificar movimientos con las dos manos, así como paréntesis, corchetes y otros símbolos con los que poder finalizar tal movimiento.

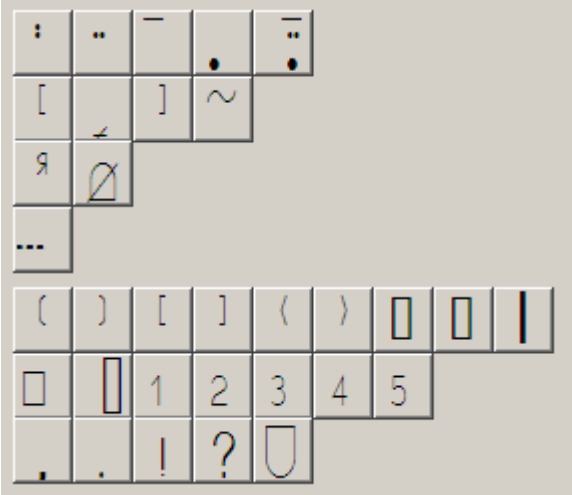


Figura 100: Pestaña *Mano Pasiva* del teclado de HamNoSys

Con el conjunto de las seis pestañas que hemos visto, se puede especificar cualquier signo en HamNoSys, únicamente haciendo click sobre el botón correspondiente. Cada vez que se aprieta uno de estos botones con los símbolos de HamNoSys, el carácter correspondiente aparece en el cuadro de texto de HamNoSys de la ventana principal. Además de aparecer el carácter, el cursor del ratón queda focalizado en el cuadro de texto, inmediatamente después del carácter introducido. Con esto se intenta facilitar el manejo de la cadena de caracteres que ya se ha introducido en el cuadro de texto de HamNoSys, pudiendo utilizar las flechas de desplazamiento o la tecla de *Retroceso* del teclado, sin tener que clickear antes sobre el cuadro de texto.

Ya hemos comentado el contenido de las seis pestañas que encontramos en la ventana secundaria *Teclado HamNoSys*, ahora sólo quedan los dos botones comunes a las seis, *Espacio* y *Retroceso*.

Estos dos botones hacen las mismas funciones que sus correspondientes teclas del teclado, introduciendo el botón *Espacio* un carácter de espacio en el cuadro de texto de HamNoSys, y borrando el carácter anterior al cursor al apretar el botón *Retroceso*.

Es importante comentar el significado del espacio en este modo de edición en HamNoSys. Al introducir un espacio, se están separando signos, que el agente animado concatenará uno detrás de otro al reproducir. Podríamos especificar la siguiente cadena de caracteres en HamNoSys:

Al reproducir esta especificación de HamNoSys, el agente animado concatenará los dos signos, primero colocando la mano activa en la parte de arriba del tronco, y posteriormente colocando la mano activa (con la misma configuración que en el anterior signo) en la parte baja del tronco. De esta manera se pretende dar la posibilidad de insertar varias sílabas para un mismo signo. El sistema permite hasta un número máximo de tres sílabas por signo, esto es, dos espacios en blanco para cada uno.

De este modo podemos editar cualquier signo en HamNoSys, haciendo uso de las utilidades correspondientes.

13.3.2. Inserción de gestos del cuerpo y la cara

El agente animado puede realizar diversos gestos, tanto con el cuerpo como con la cara, mientras reproduce cualquier signo.

Podemos introducir cualquiera de estos gestos en cualquier modo de edición, aunque siempre para reproducir un signo debemos tener la descripción de la parte manual en HamNoSys. Para poder introducir los gestos debemos abrir la ventana correspondiente, a partir del botón Añadir, del campo de Expresividad Cara y Cuerpo.

Las especificaciones tanto de SEA como de HamNoSys no contemplan la posibilidad de insertar gestos de la cara y del cuerpo. Debido a este hecho necesitamos de este diálogo auxiliar para poder insertarlos, indicando al agente animado los tipos de gestos que debe representar.

Como podemos apreciar, esta ventana contiene tres pestañas y seis botones comunes a las mismas.

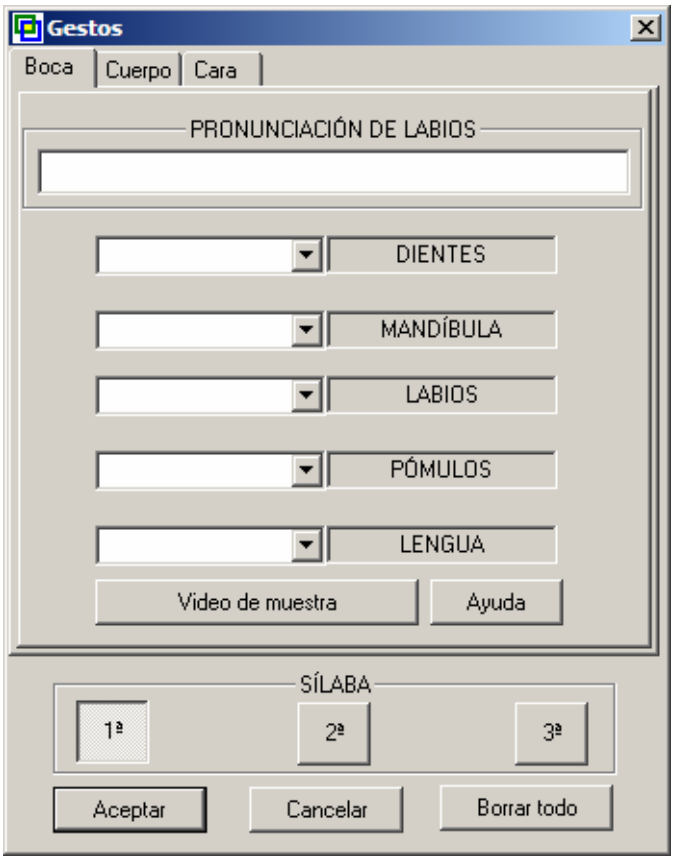


Figura 101: Diálogo para la inserción de gestos de la cara y del cuerpo

La primera pestaña, Boca, contiene las utilidades necesarias para introducir gestos referentes a la boca, incluyendo la pronunciación de cualquier palabra en castellano. En el campo *PRONUNCIACIÓN DE LABIOS* podemos escribir cualquier palabra, para que el agente animado la pronuncie con los labios a la vez que realiza el signo especificado. El programa realiza internamente una conversión de los grafemas del castellano a los alófonos utilizando el estándar SAMPA, pudiendo así insertar cualquier pronunciación de sonidos del castellano.

Además de este campo para la pronunciación de palabras, en esta pestaña hay otros cinco campos con sus correspondientes menús desplegables, mediante los cuales podemos insertar diversos gestos. Para los *DIENTES* podemos escoger hasta nueve gestos distintos, uno para la *MANDÍBULA*, treinta y uno para los *LABIOS*, trece para los *PÓMULOS*, y diecisiete para la *LENGUA*. En cuanto a los gestos de esta pestaña, sólo podremos tener insertado un gesto en cada sílaba, lo que quiere decir que no se puede tener insertado a la vez un gesto de *DIENTES*, por ejemplo, a la vez que un gesto de *PRONUNCIACIÓN DE LABIOS* (en la misma sílaba).

Existen en esta pestaña dos botones especiales, *Video de muestra* y *Ayuda*. Estos dos botones sirven para comprobar qué tipo de signo corresponde a los códigos que aparecen en las listas desplegables. Por ejemplo, en el caso de la lista desplegable *LABIOS*, existen hasta treinta y un tipos de gestos distintos. Estos están especificados en la lista con códigos numéricos, del L01 al L31. Para poder comprobar el tipo de gesto al que se refieren, podemos hacer uso de los videos de muestra, o por el contrario de imágenes explicativas.

Para elegir los videos de muestra debemos hacer uso del botón *Video de muestra*.

Figura 102: Botón *Video de muestra*

Al pulsar este botón, aparecerá un video demostración del último gesto seleccionado, ya sea de *DIENTES, MANDÍBULA, LABIOS, PÓMULOS* o *LENGUA*.

Figura 103: Video demostración del gesto L10

Si por el contrario queremos comprobar el tipo de gesto a insertar mediante la visualización de imágenes explicativas, debemos hacer uso del botón *Ayuda*.

Figura 104: Botón *Ayuda*

Una vez pulsado el botón, se abrirá un documento de Word, *ayuda_gestos.doc*, donde aparecen las imágenes explicativas de todos los gestos disponibles en la pestaña *Boca*.

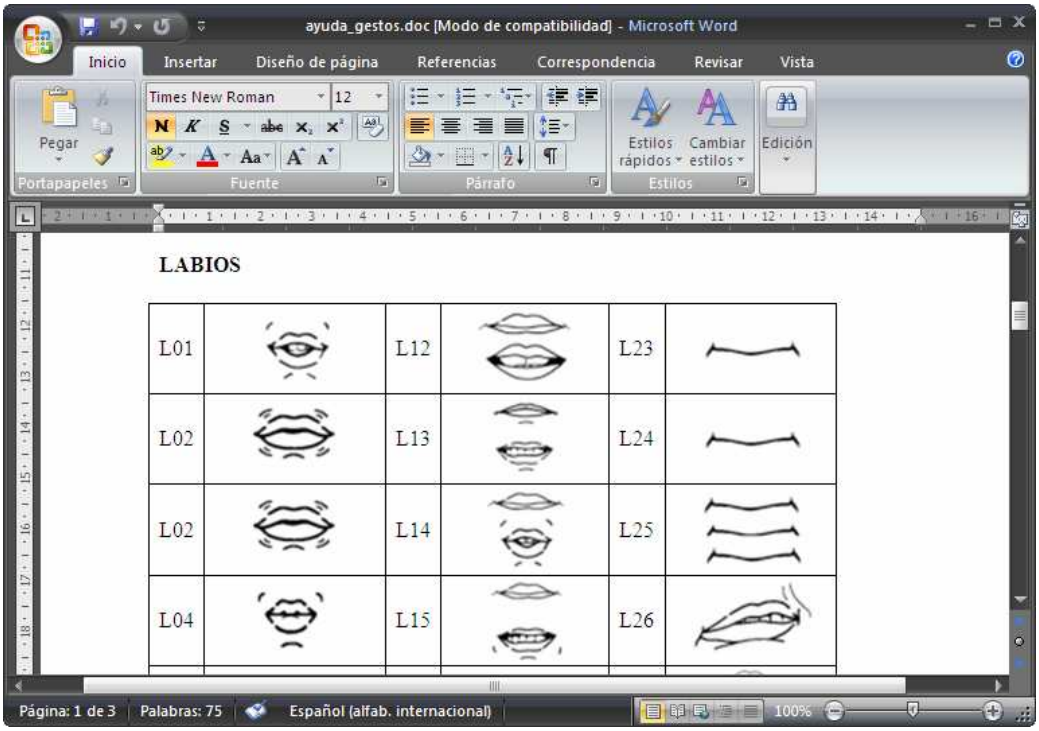


Figura 105: Documento Word con las imágenes explicativas de los gestos de la pestaña *Boca*

La siguiente pestaña, *Cuerpo*, contiene los botones mediante los cuales podremos introducir gestos realizados con los hombros. Existen tres posibles movimientos expresados con los hombros. Podremos subirlos, echarlos hacia delante, y subirlos y a continuación bajarlos. También podemos especificar qué hombro es el que realizará el movimiento, pudiendo elegir entre el izquierdo, el derecho, o ambos.

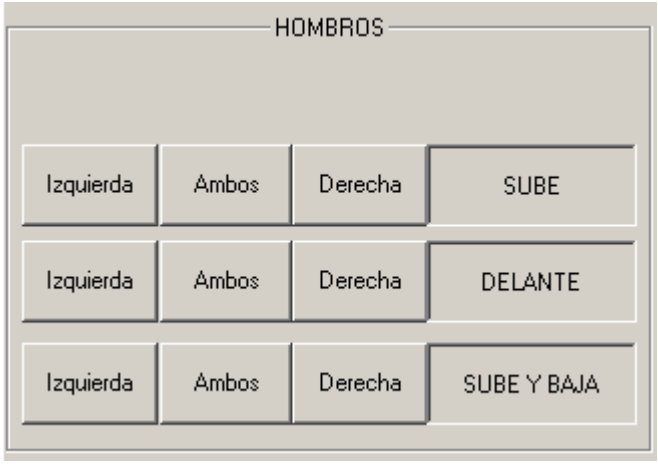


Figura 106: Pestaña *Cuerpo* del diálogo de inserción de gestos

En esta pestaña, al igual que ocurría en la pestaña *Boca*, únicamente podremos tener insertado un gesto por sílaba.

Mediante la última pestaña, *Cara*, podremos introducir gestos con la mirada, las cejas, los párpados y la nariz. Para la mirada encontramos cuatro

opciones, hacia arriba, hacia abajo, hacia la derecha y hacia la izquierda, todas ellas con su correspondiente botón. Para las cejas, párpados y nariz, podremos elegir entre varias opciones en su correspondiente menú desplegable, habiendo un total de cuatro gestos a insertar para *CEJAS*, cinco para *PÁRPADOS,* y tres para *NARIZ.*

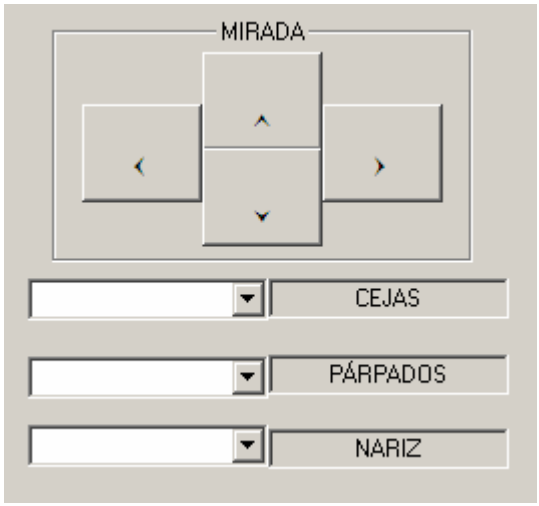


Figura 107: Pestaña *Cara* del diálogo de inserción de gestos

Es muy importante conocer el funcionamiento de las utilidades comunes a las tres pestañas, ya que para cualquier inserción de gestos debemos hacer uso de ellas.

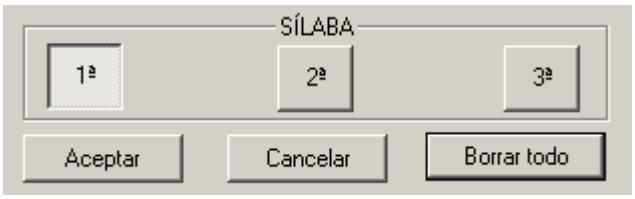


Figura 108: Botones comunes del diálogo de inserción de gestos

Como podemos ver en la figura, hay tres botones referentes a tres sílabas diferentes que se pueden incluir. En Hamnosys se diferencian las sílabas con espacios en blanco. Esta sílaba se corresponde con la separación del espacio comentada anteriormente, donde el agente animado iba concatenando un signo (sílaba) tras otro. Al insertar un gesto, primero debemos elegirlo en la pestaña adecuada, y a continuación pulsar al botón de *Aceptar*, y dependiendo de la sílaba que esté pulsada, el gesto elegido se insertará en la primera, segunda o tercera sílaba del signo. Por defecto, al abrir la ventana de gestos, la sílaba que está pulsada es la primera, por lo que si queremos insertar un gesto en la segunda o en la tercera, debemos pulsar el botón correspondiente.

El botón *Cancelar* cierra la ventana de gestos, descartando cualquier selección de gesto realizada. Con el botón *Borrar todo*, eliminamos todas las inserciones realizadas, en cualquiera de las tres sílabas.

Dentro de la ventana principal hay un campo en el que van apareciendo todos los gestos que hemos ido insertando, y que se quedan guardados hasta

que sean borrados (mediante el botón *Borrar todo* del diálogo de inserción de gestos). Este campo se encuentra dentro del de *Expresividad Cara y Cuerpo*, debajo del cuadro de texto de HamNoSys. Cada vez que pulsamos una nueva sílaba en el diálogo de inserción de gestos, este campo cambia, apareciendo los gestos actualmente guardados (validados mediante la pulsación del botón *Aceptar*) en la sílaba pulsada. Por defecto, cuando no hay ningún gesto insertado, el campo tiene el siguiente aspecto:

Figura 109: Gestos guardados en la primera sílaba

Si queremos insertar en la segunda sílaba, por ejemplo, un gesto referente al hombro y la pronunciación de la palabra “hola”, al pulsar el botón *Aceptar* después de haber seleccionado la segunda sílaba, debe aparecer en la ventana principal, en el campo referente a los gestos, lo siguiente:

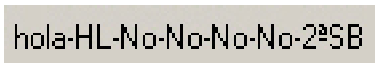


Figura 110: Gestos guardados en la segunda sílaba

En cualquier momento podemos tener gestos insertados en las tres sílabas, aunque únicamente tengamos dos o una sílaba especificada en HamNoSys. El agente animado únicamente representará aquellos gestos en los que sí exista alguna sílaba especificada en HamNoSys (parte manual).

13.3.3. Modo de edición en SEA

El otro modo de edición de signos en la herramienta es la especificación de signos en SEA, lo que podremos hacer escribiendo los signos en el cuadro de texto correspondiente, que se encuentra encima del de edición en HamNoSys.

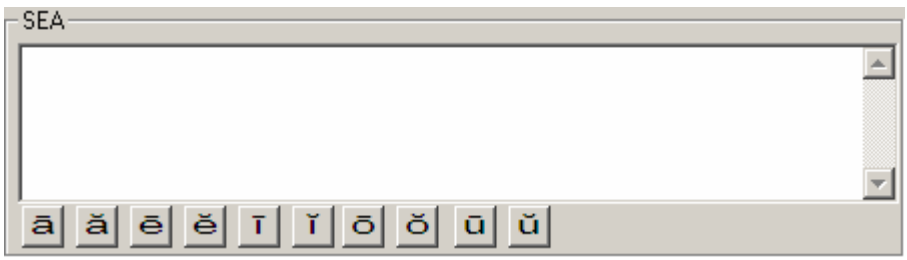


Figura 111: Cuadro de edición de signos en SEA

Mediante el teclado corriente, puesto que la especificación de SEA utiliza los caracteres del alfabeto convencional, escribiremos en el cuadro de texto el signo que queramos representar por el agente animado. Como ya hemos comentado anteriormente, el signo que escribamos en SEA no puede ser representado directamente, ya que habrá que convertirlo antes a HamNoSys. Por tanto, una vez escrito el signo en SEA, debemos pulsar el botón *Convertir SEA-HamNoSys*, que se encuentra debajo del cuadro de texto de edición de SEA, y aparecerá su conversión en el cuadro de texto de HamNoSys.

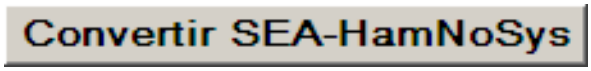


Figura 112: Botón *Convertir SEA-HamNoSys*

Los dos estándares son bastante diferentes, por lo que no se puede conseguir una conversión perfecta de SEA a HamNoSys. Aún así, en el 70% de los casos, aproximadamente, la conversión es bastante aceptable. En el 30% restante es necesario realizar ciertos retoques manuales (mediante el teclado de HamNoSys).

Una manera alternativa de traducir los signos de SEA a HamNoSys es, en vez de pulsar el botón *Convertir SEA-HamNoSys*, pulsar la tecla *Enter* del teclado cuando el ratón se encuentra en el cuadro de texto de SEA.

Dentro de la especificación de SEA, un signo puede contener varias sílabas, que se corresponden con distintos movimientos, tanto de la mano pasiva, la activa o ambas. A pesar de ser sílabas diferentes, todas ellas pertenecen a una misma unidad, el signo. Para separar estas sílabas la notación escogida es la del guión, pudiendo ser el siguiente ejemplo la especificación de un signo en SEA compuesto por dos sílabas:

re-v

Una vez traducido el signo a HamNoSys, podremos representarlo en el agente animado, al igual que haríamos si hubiéramos escrito directamente el signo en HamNoSys. En HamNoSys las sílabas se separan por un espacio en blanco.

La especificación de SEA se basa en la utilización del alfabeto convencional, por lo que se puede utilizar el teclado corriente. Sin embargo, hay ciertas configuraciones que necesitan caracteres con tildes poco usuales, que no pueden ser escritas con el teclado convencional. Para ello, debajo del cuadro de texto de SEA, encontramos los botones correspondientes a estos caracteres, que al pulsarlos se insertarán directamente.

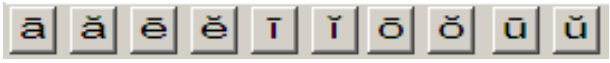
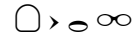


Figura 113: Botones para la inserción de caracteres especiales en SEA

13.3.4. Modo de edición en SiGML

El lenguaje SiGML está basado en XML, y es el que recibe el agente animado para representar un signo. Esto quiere decir que cada signo que va a ser representado por el agente animado debe estar codificado en SiGML, único lenguaje que entiende el avatar. Este lenguaje está íntimamente ligado con HamNoSys, ya que cada símbolo de éste se corresponde con una etiqueta de SiGML. Vamos a poner un ejemplo. Supongamos que tenemos el siguiente signo escrito en HamNoSys (parte manual):



El signo se compone de cuatro símbolos, el primero es de la configuración de la mano (mano abierta), el segundo indica la dirección de la mano (hacia la derecha), el tercero indica la orientación de la palma (palma hacia arriba) y el cuarto se refiere a la localización (en los ojos). Estos cuatro símbolos se corresponden con cuatro etiquetas SiGML. El signo entero codificado en este lenguaje sería el siguiente:

```
<sigml>
    <hns_sign gloss="$PROD">
        <hamnosys_nonmanual>
        </hamnosys_nonmanual>
        <hamnosys_manual>
            <hamflathand/>
            <hamextfingerr/>
            <hampalmu/>
            <hameyes/>
        </hamnosys_manual>
    </hns_sign>
</sigml>
```

Tabla 32: Ejemplo de codificación SiGML

Las cuatro etiquetas referentes a los símbolos de HamNoSys se encuentran entre las etiquetas *<hamnosys_manual>* y *</hamnosys_manual>*. Especificado el signo de esta manera es como lo entiende el agente animado y entonces podría ser representado por éste. La codificación correspondiente a la inserción de los gestos de la cara y del cuerpo también se realiza mediante etiquetas, en este caso entre las etiquetas *<hamnosys_nonmanual>* y *</hamnosys_nonmanual>*.

En la ventana principal, debajo del cuadro de texto de la expresividad de cara y cuerpo, encontramos los botones con los que podemos manipular los signos en SiGML.

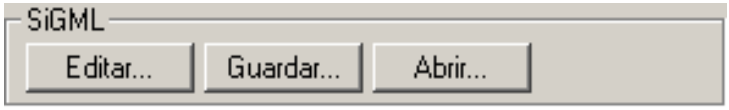


Figura 114: Botones para la edición de signos en SiGML

Mediante el botón *Abrir…* podemos cargar cualquier archivo de texto que ya contenga la especificación de un signo, para así directamente poder reproducirlo (al abrir el archivo el agente animado lo reproduce) sin haberlo insertado previamente en los cuadros de texto de SEA o de HamNoSys.

Con el botón *Editar…* abrimos una ventana de edición de texto, que en su defecto estará en blanco (a no ser que previamente se haya cargado algún

archivo o hayamos insertado algún símbolo en el cuadro de texto de HamNoSys). En esta ventana podremos editar cualquier signo en el lenguaje SiGML, empezando de cero o retocando el que esté cargado. Como ya hemos comentado, al insertar símbolos en el cuadro de texto de HamNoSys para especificar un signo, ya tendremos su correspondencia en SiGML, por lo que en la ventana de edición ya encontraremos sus etiquetas, que podremos retocar como queramos.

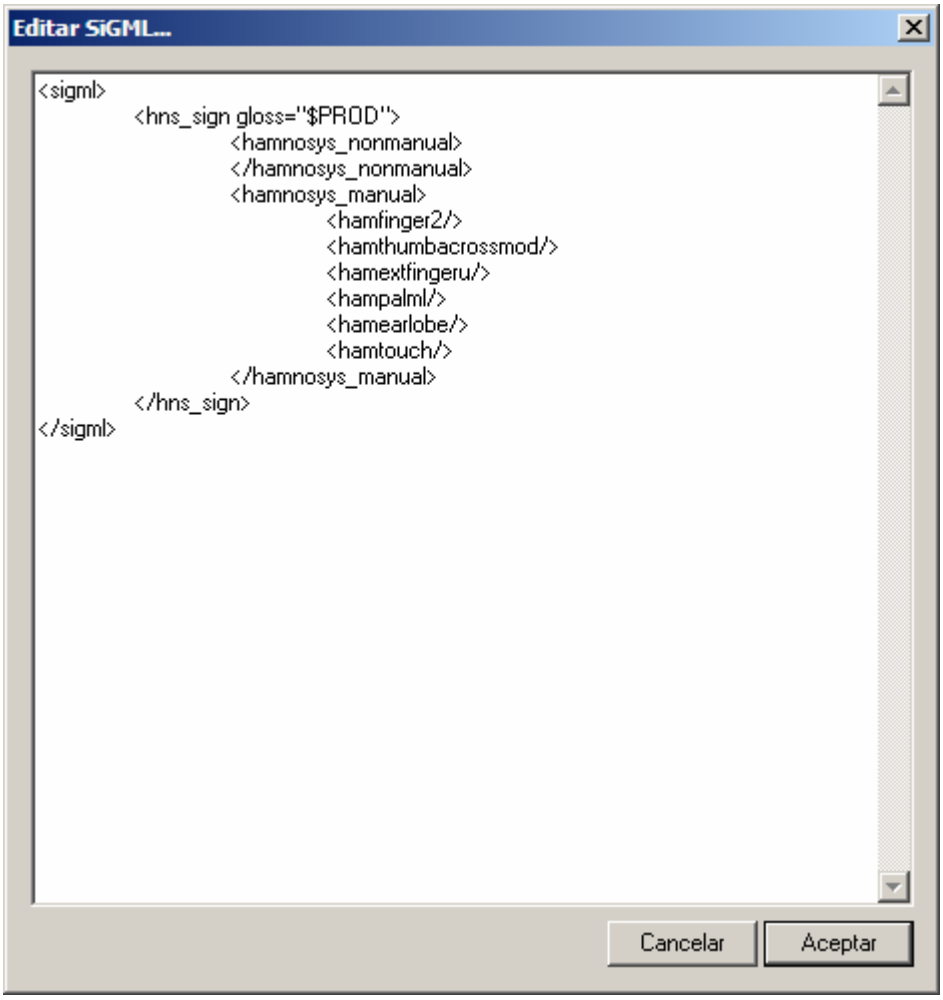


Figura 115: Diálogo de edición de signos en SiGML

Si realizamos cualquier cambio en la ventana de edición y posteriormente pulsamos el botón *Aceptar*, esta ventana se cerrará y los cambios se quedarán guardados. Esto quiere decir que si volvemos a la ventana de edición desde la principal, veremos los cambios reflejados. También podemos comprobarlo en la ventana principal, ya que al haber cambiado las etiquetas en SiGML, habrán cambiado los símbolos HamNoSys a los que representan, lo que hace que el cuadro de texto de HamNoSys se modifique.

Dentro del campo *SiGML* de la ventana principal, también encontramos el botón *Guardar…*, con el que podremos guardar en un archivo de texto la especificación en SiGML del signo actual, ya sea por edición directa en SiGML, o por correspondencia con los símbolos del cuadro de texto de HamNoSys. Si

dentro de la ventana de edición pulsamos el botón *Cancelar*, esto se indica en el cuadro de estado, informando de que los cambios no serán aplicados.

13.3.5. Representación y estado

Debajo del campo de SiGML en la ventana principal encontramos el campo de estado, donde se va indicando el estado en el que se encuentra el programa. Nos da cierta información acerca de la reproducción de los signos y de la edición en SiGML de éstos.

Figura 116: Cuadro de estado

Una vez lanzado el programa (e inicializado el avatar), el estado por defecto será el de *Avatar cargado.*

La información que aparece en el campo de estado más importante es la referente a la representación del signo por el agente animado. Podemos representar un signo de diversas maneras. Ya hemos comentado que al cargar un archivo de texto que contiene un signo en SiGML, directamente se representa. También podemos pulsar la tecla *Enter* cuando nos encontramos en el cuadro de texto de HamNoSys. La opción que nos queda es pulsar el botón *Representar*, que encontramos en la ventana principal a la izquierda del campo de SiGML.

Figura 117: Botón *Representar*

Al realizar cualquiera de estas operaciones, el estado corresponderá a *Reproduciendo…*. Una vez reproducido el signo por el agente animado, el estado será el de *Reproducción correcta. El avatar acepta el código SiGML.*, lo que querrá decir que el signo está correctamente especificado. Si esto no fuera así, el estado se quedaría en *Reproduciendo…*, por lo que habría que corregir el signo para que el agente animado pudiera representarlo.

13.4. Principales problemas

En la tabla siguiente podemos ver los principales problemas que pueden aparecer en la aplicación, así como sus causas y su posible solución.

PROBLEMAS	CAUSAS	SOLUCIÓN
El agente animado no aparece	No se ha instalado el control ActiveX VGuido o la Máquina Virtual	Instalar los dos elementos. Mirar

	Java	apartado de instalación
Los símbolos de HamNoSys o de SEA no apareen	Las fuentes no están instaladas	Instalar en FONTS los archivos *hamnosys4.ttf* y *sea.ttf*
No se ven los videos	No se ha instalado el programa QuickTime	Instalar el programa Mirar apartado de instalación
Al intentar representar, el agente animado no hace nada	La especificación de HamNoSys es incorrecta	Corregir la especificación en HamNoSys

Tabla 33: Resolución de problemas

14. ANEXO III: Contenido del DVD

Este apartado describe el contenido del DVD que se adjunta con este informe. Este contenido está organizado en 3 carpetas: contenidos digitales, documentos y programas generados durante el proyecto.

14.1. Contenidos Digitales Generados

El documento Descripcion_BD_frases.doc describe con detalle las dos bases dedatos de frases generadas para los dos entornos:

- BD de frases para la renovación del DNI
- BD de frases para la renovación del carné de conducir.

La descripción de los signos generados se incluye en el documento base_de_datos_signos.doc.

- Signos para la renovación del DNI y del carné de conducir

14.2. Documentos

Los principales documentos son los siguientes:

- Informe final del proyecto: en este informe se incluyen todos los trabajos realizados, la evaluación y los resultados obtenidos. El fichero es el informe_final_del_proyecto.doc.
- Presentaciones de las reuniones del proyecto : Reunion1_v3.ppt, Reunion2_v1.ppt y Reunion3_v1.ppt
- Fotos y vídeos realizados en la DGT de Toledo.
- Curso de HamNoSys
    - Transparencias del curso: HamNoSys_v2.ppt.
    - Documentación adicional: Introducción a HamNosys_castellano.doc.
- Reglas de etiquetado de glosas: informe_etiquetado_glosas.doc.
- Documento sobre la evaluación de los sistemas: informe_evaluación_dgt.doc
- Manual de usuario del sistema de traducción de voz a LSE: manual_usuario_traductor_voz_a_LSE.doc
- Manual de usuario del Editor de Signos: manual_usuario_editor_signos.doc

14.3. Programas de ordenador

- Programas necesarios:
    - Quicktime
    - Agente animado
    - Máquina Virtual Java

    - Microsoft SAPI 5.1
- Programas desarrollados en este proyecto:
    - Sistema de traducción de voz a LSE para la renovación del DNI.
    - Sistema de traducción de voz a LSE para la renovación del carné de conducir en la DGT.
    - Editor de signos.
    - Fuentes sea y HamNoSys.

15. BIBLIOGRAFÍA

[1] A. Herrero, "*Escritura Alfabética de la Lengua de Signos Española*", Publicaciones de la Universidad de Alicante, 2003.

[2] Alon Lavie, "*Machine Translation Overview*", Language Technologies Institute, Carnegie Mellon University, LTI Immigration Course, 2008.

[3] Base de datos MSDN de Microsoft. **http://msdn2.microsoft.com**

[4] Ceballos Sierra, Francisco Javier. "*Visual C++: aplicaciones para Win32*". Ra-ma. 2003.

[5] Descripción de HamNoSys. Página web de la Universidad de Hamburgo: **http://www.sign-lang.uni-hamburg.de/projects/hamnosys.html**

[6] "Diccionario Normativo de la Lengua de Signos Española (DILSE III)". Fundación CNSE. Formato DVD. 2008.

[7] Fernández Gaspar, Isabel. "Mejora y ampliación de un sistema de traducción de texto a Lengua de signos". PFC ETSIT-UPM.

[8] Ibáñez León, Eloísa. "Sistema de traducción de lenguaje natural a signos gestuales". PFC ETSIT-UPM.

[9] Ibáñez, E., Huerta, A., San-Segundo, R., D'Haro, L.F., Fernández, F., Barra, R. "*Prototipo de traducción de voz a Lengua de Signos Española*". IV Jornadas en Tecnologías del Habla. 8-10 Noviembre 2006. Zaragoza.

[10] J. Abaitua, "*Introducción a la traducción automática*", Grupo DELi, Universidad de Deusto. 2001.

[11] J. Cassell. "Embodied Conversational Agents: Representation and Intelligence in User Interfaces". AI Magazine. Winter 2001.

[12] Kernighan, Brian W. "*The C programming language*". Prentice-Hall. 1998.

[13] M.A. Rodríguez González, *"Lengua de Signos"*, Tesis Doctoral 1996. CNSE y Fundación ONCE. Biblioteca Virtual Miguel de Cervantes.

[14] Página del proyecto eSIGN: **http://www.sign-lang.uni-hamburg.de/esign/**

[15] Página web personal de W. J. Hutchins, donde muestran ciertas publicaciones, así como referencias sobre traducción automática: **http://www.hutchinsweb.me.uk/**

[16] Pinedo Peydro, F. J. "*Diccionario de Lengua de Signos Española*". Fundación CNSE. 2000.