HanaNLG: Sistema de generación de lenguaje híbrido y flexible

HanaNLG: A Flexible Hybrid System for Natural Language Generation

Resumen de la aplicación: HanaNLG (Hybrid surfAce realisatioN Approach for Natural Language Generation) es un sistema híbrido para la fase realización capaz de generar automáticamente texto que es fácilmente adaptable a diferentes géneros, dominios y lenguajes. HanaNLG es híbrido porque se basa en el uso de recursos lingüísticos así como en información estadística, a través del uso de Modelos de Lenguaje Factorizados, para construir la salida final. Para generar lenguaje, el sistema propuesto hace uso de estrategias de *over-generation y ranking*, donde primero se genera un conjunto de frases candidatas para después realizar un ranking donde se seleccione una frase en base a un criterio definido, en nuestro caso, su probabilidad. Además, dado que HanaNLG solo está enfocado en la fase de realización, no tenemos información de los procesos de macroplanificación y microplanificación, para poder guiar la generación en base a un tema concreto, palabras, dominio, etc. . Es por ello, que proponemos el concepto de característica semilla. Estas características semillas pueden considerarse objetos abstractos (por ejemplo, fonemas, sentimientos, polaridades, etc.) que guiarán el proceso de generación en relación al vocabulario que la frase generada deba contener. Por consiguiente, el tipo de textos generados por HanaNLG puede ser adaptado a diferentes dominios y también a diferentes objetivos comunicativos (por ejemplo, generación automática de resúmenes). Asímismo, dada la naturaleza de los recursos y técnicas empleadas, nuestro sistema también es fácilmente adaptable a diferentes géneros, dominios y lenguajes. Una descripción más detallada del método completo se puede encontrar en [Barros19, Barros17]

Summary: HanaNLG (Hybrid surfAce realisatioN Approach for Natural Language Generation) is a hybrid system for the surface realisation stage capable of automatically generating text which is easily adaptable to different genres, domains and language. HanaNLG is hybrid because the final text is created based on the use of linguistic resources in conjunction with statistical information, through the use of factored language models. In order to generate language, this approach makes use of overgeneration and ranking strategies, where a set of candidate sentences is first generated and then a ranking is performed to select a sentence based on a defined criterion, which, in our case, is the sentence probability. In addition, since HanaNLG is only focused on the surface realisation stage, there is not any information about the macroplanning and microplanning processes, so, to guide the generation on the basis of a specific theme, words, domain, etc., we propose the concept of seed feature. These seed features can be considered as abstract objects (e.g., phonemes, emotions, polarities, etc.) which will guide the generation process in relation with the vocabulary that the generated sentence must contain. Therefore, the text generated by HanaNLG can be adapted to different domains and also to different communicative goals (e.g., automatic summarisation). Likewise, given the nature of the resources and techniques employed, our system is also easily adaptable to different genres, domains and language. A detailed description of the complete approach can be found at [Barros19, Barros17].

Especificaciones técnicas/ Technical Specifications

Lenguaje de programación/ Development Language: Java

Entorno Operativo/ Operating Environment: Linux

Versión/Version: 1.0

Estructura de ficheros/ Files' structure:

Directorio/Folder -> SeedF3:

- README.md
- build.xml
- manifest.mf

Directorio/Folder -> SeedF3/nbproject:

- build-impl.xml
- genfiles.properties
- project.properties
- project.xml
- **Subdirectorio/Subfolder -> configs**
  - Sentiment.properties
  - Test.properties
  - Timelines.properties

Directorio/Folder -> SeedF3/src/seedf3:

- SeedF3.java
- **Subdirectorio/Subfolder -> Components**
  - NP.java
  - NPType.java
  - PP.java
  - S.java
  - SObject.java
  - SObjectType.java
  - SentenceType.java
  - Subject.java
  - SubjectType.java
  - Word.java
  - WordInfo.java
- **Subdirectorio/Subfolder -> Features**
  - Letter.java
  - Phoneme.java
  - Polarity.java
  - SeedFeature.java
  - Sentiment.java
  - VImpW.java
- **Subdirectorio/Subfolder -> Generation**
  - BagW.java
  - Gen.java
- **Subdirectorio/Subfolder -> Test**
  - TestTools.java
- **Subdirectorio/Subfolder -> Tools**
  - AuxFunctions.java
  - Inflect.java
  - Lexicon.java
  - LoadFunctions.java
  - Postproc.java
  - Preproc.java
  - Rank.java
  - VNHandler.java
  - WNHandler.java

Directorio/Folder -> SeedF3/files:

- **Subdirectorio/Subfolder -> features**
  - feng
  - fesp
  - imp
  - negative-words.txt
  - opt2_h2.len3.s25.cuento_67_en.plmInv
  - positive-words.txt
  - senticon.en.xml
  - union-emosenticnet-emolex-complete-only1emotion.csv
- **Subdirectorio/Subfolder -> lexicons/freeling**
  - **Subdirectorio/Subfolder -> enFull**
    - adjs
    - adjs.comp
    - adv
    - adv.comp
    - contraccions

        - extra
        - interj
        - noms
        - tanc
        - verbs
        - verbs.aux
    - **Subdirectorio/Subfolder -> es**
        - MM.adj
        - MM.adv
        - MM.int
        - MM.nom
        - MM.tanc
        - MM.vaux
        - MM.verb
- **Subdirectorio/Subfolder -> libraries**
    - edu.mit.jverbnet-1.2.0.jar
    - lttoolbox.jar
    - **Subdirectorio/Subfolder -> JWI 2.4**
        - edu.mit.jwi_2.4.0.jar
        - edu.mit.jwi_2.4.0_javadoc.zip
        - edu.mit.jwi_2.4.0_src.zip
    - **Subdirectorio/Subfolder -> VNdict (274 files)**
    - **Subdirectorio/Subfolder -> WNdict30 (24 files)**
- **Subdirectorio/Subfolder -> restrictions**
    - noframelist
    - not_desirable_verbs
    - stopeng

Requerimientos/Requirements:

- Java 1.8 o superior instalado
- Freeling 4.0 o superior instalado
- SRILM 1.7.3 o superior instalado
- Netbeans

Instalación/Installing: Una vez descargado el sistema, se debe compilar el código para generar los binarios, para ello se ha de importar el proyecto en Netbeans y se ha de compilar mediante la función `build` de Netbeans. Esto permite que se genere un fichero JAR en la carpeta `dist/`, el cual será utilizado para ejecutar este sistema.

Antes de comenzar a utilizar el sistema, es necesario asegurarse de que Freeling y SRLIM están instalados y funcionan correctamente.

Una vez está todo comprobado, el sistema está listo para ser utilizado.

Ejecución/Run: Para que el sistema pueda funcionar correctamente, es necesario disponer de un corpus previamente etiquetado y entrenar los modelos de lenguaje en base a este corpus etiquetado. Por ello primero ejecutaremos el sistema con la opción `-train` indicando el directorio donde se encuentra el corpus a etiquetar en texto sin formato, y también el directorio de salida donde se almacenará el corpus etiquetado junto con los modelos de lenguaje. Esto nos permitirá tener los ficheros preparados para poder generar texto.

En el caso de tener de antemano los ficheros anteriormente mencionados, el paso anterior no sería necesario.

Una vez están disponibles estos ficheros, se ejecuta el sistema con la opción `-gen` indicando el tipo de característica a emplear (fonema, polaridad, keywords), los ficheros asociados a esta característica (para poder detectarla), el nivel de abstracción (si se vana

emplear palabras, lemas o synsets para generar las frases), el número de frases a generar, si las frases van a estar relacionadas entre sí, el corpus a emplear y el directorio de salida donde se almacenarán las frases generadas.

Dependencias/Dependencies:
El listado completo de dependencias se incluyen a continuación, en el cual las librerías propias del sistema están marcadas con un *:

- edu.mit.jwi_2.4.0.jar
- edu.mit.jwi_2.4.0_javadoc.zip
- edu.mit.jwi_2.4.0_src.zip
- lttoolbox.jar
- edu.mit.jverbnet-1.2.0.jar
- taggerdemo_1.0.0.jar *

Referencias/References

[Barros19] Barros, C. (2019). *Proposal of a Hybrid Approach for Natural Language Generation and its Application to Human Language Technologies* (Doctoral thesis, University of Alicante). Retrieved from http://hdl.handle.net/10045/100145

[Barros17] Barros, C. and Lloret, E. . 2017. *A multilingual multi-domain data-to-text natural language generation approach*. Procesamiento del Lenguaje Natural. 58, pp 45-52. Retrieved from http://dx.doi.org/10.26342/2017-58-5411