Realização Superficial baseada em Regras

Douglas F. P. da Silva Junior, Eder M. de Novais, Ivandré Paraboni

[1]School of Arts, Sciences and Humanities (EACH)
University of São Paulo (USP)
Av. Arlindo Bettio, 1000 - São Paulo, Brasil

douglas.fernandes.silva, eder.novais, ivandre@usp.br

Abstract. *This paper presents a surface realisation engine for the Brazilian Portuguese language. The system is presented as a library of functions, and it has been applied to the generation of newspapers headlines.*

Resumo. *Este trabalho apresenta um sistema de realização superficial para o Português brasileiro. O sistema é apresentado na forma de uma biblioteca de funções, e aplicado à tarefa de geração de manchetes de jornal.*

1. Introdução

Em sistemas de geração de língua natural (GLN), a realização superficial é a tarefa computacional de mapear uma representação abstrata do texto - frequentemente já contendo algum tipo de informação estrutural - para uma representação linear em língua natural [Reiter 2007]. A entrada de um módulo de realização textual é algum tipo de especificação abstrata da sentença, e a saída é um *string* em língua natural.[1]

Este trabalho enfoca a implementação e avaliação de um protótipo de sistema de realização superficial para o Português brasileiro. O sistema proposto é apresentado na forma de uma biblioteca de métodos de geração de sintagmas e sentenças completas, os quais podem ser acoplados a uma aplicação subjacente que necessite expressar dados não linguísticos em formato textual.

2. O Sistema Proposto

O presente trabalho pode ser visto como um equivalente para língua portuguesa do sistema *SimpleNLG* [Gatt and Reiter 2009] para o inglês. Seguindo [Gatt and Reiter 2009], enfocamos apenas o problema da realização superficial operacional[2], pressupondo que a escolha lexical e a seleção da estrutura da frase já tenham sido decididas. Restam assim as tarefas de completar as informações de entrada (se esta for subespecificada), estabelecer a concordância entre os termos da sentença e escolher a ordem linear destes[Malouf 2000, Mitchell 2009].

Assim como em [Gatt and Reiter 2009], o presente trabalho é disponibilizado na forma de uma biblioteca de métodos JAVA para construção de sentenças. Esta biblioteca foi projetada para ser inserida no código da aplicação subjacente e, em ambos os sistemas, exige um certo grau de conhecimento de programação para sua utilização.

[1]Ou seja, ao contrário da tarefa de *interpretação* de língua natural [Cuevas and Paraboni 2008].

[2]Problemas de geração de textos em português com base em conhecimento profundo são abordados em [Paraboni 2003, Paraboni et al. 2006, de Lucena et al. 2010]

Proceedings of the 9th Brazilian Symposium in Information and Human Language Technology, pages 148–152,
Fortaleza, CE, Brazil, October 21–23, 2013.

Como forma de facilitar a adaptação do sistema a novas aplicações, foram implementadas duas modalidades de uso: uma ao estilo utilizado em [Gatt and Reiter 2009], em que a aplicação subjacente realiza uma série de chamadas incrementais aos métodos de construção da sentença, e uma modalidade de chamada única, na qual é fornecida uma árvore da sentença a ser gerada, semelhante ao padrão dito 'superficial' adotado no *Generation Challenges 2011 Surface Realisation Shared Task*[3].

Estas modalidades são discutidas separadamente nas seções a seguir com uso de exemplos simplificados por questões de clareza. Estes exemplos ilustram apenas algumas das diversas opções e recursos implementados para geração de texto.

2.1. Geração Incremental

Na sua forma mais típica de uso, o sistema é invocado mediante uma série de chamadas a métodos de construção de sintagmas e outros componentes da sentença. A criação de uma sentença completa consiste em invocar os métodos necessários para construção dos componentes individuais da estrutura, e então combiná-los em uma frase de acordo com um template pré-definido (e.g., voz ativa, passiva, imperativo etc.)

A classe *SintagmaNominal* permite a criação de diversos tipos de sintagma nominal (SN). A criação de um SN é feita invocando-se seu método construtor e especificando-se seus parâmetros como *Lemma* (i.e., a forma-base do núcleo do SN), *DetLemma* (o núcleo do termo determinante), gênero, número etc. Além disso, é possível utilizar Sintagmas adjetivais (classe *SintagmaA*) e preposicionais (classe *SintagmaP*) para modificar um SN. No exemplo a seguir, é criado um SN com determinante do tipo possessivo 'as lindas crianças da escola':

SintagmaNominal sn_1 = SintagmaN.none();
sn_1.setDetLemma("o");
sn_1.setLemma("escola");

SintagmaP sp = new SintagmaP();
sp.setPreposicao("de");
sp.setConsequente(sn_1);

SintagmaA sa_1 = new SintagmaA();
sa_1.setAdjetivo("lindo");

SintagmaNominal sn_2 = SintagmaN.def();
sn_2.setLemma("criança");
sn_2.setDetLemma("o");
sn_2.setModificadorSintagmaAdjetivo(sa_1, Ordem.antes());
sn_2.SetPPlist(sp);

A classe *SintagmaVerbal* permite a criação de sintagmas verbais (SV) de diversos tipos, que podem ainda ser modificados por sintagmas adverbiais (classe *SintagmaAdverbial*). O exemplo a seguir ilustra a criação da cadeia 'estão sorrindo alegremente'.

SintagmaVerbal sv = SintagmaV.VGer();
sv.setVerboPrincipal("estar");
sv.setVerboSegundario("sorrir");
sv.setParametroInt(Genero.Fem(),Numero.Plural(), Pessoa.Terceira(), Modo.Indic(), Tempo.Presente());

[3]http://www.nltg.brighton.ac.uk/research/sr-task/

SintagmaAdverbial $sadv$ = new SintagmaAdverbial();
$sadv$.setAdverbio("alegremente");
sv.setModificadorSintagmaAdverbial($sadv$, Ordem.depois());

Finalmente, SNs e SVs podem ser combinados em uma sentença completa assumindo uma de quatro formas (ou templates) possíveis: voz ativa, voz passiva, imperativo e sujeito oculto. Um exemplo de geração de 'As lindas crianças da escola estão sorrindo alegremente' é apresentado a seguir.

vozAtiva $frase$ = new vozAtiva();
$frase$.setSN1(sn_1);
$frase$.setSN2(sn_2);
$frase$.setSV(sv);

2.2. Geração a partir de Árvore Subespecificada

A segunda modalidade de uso do sistema proposto é adequada para situações nas quais a informação a ser gerada já se encontra parcialmente disponível em forma textual. Problemas desta natureza, chamados de geração Texto-para-Texto, têm sido abordados, por exemplo, em [Belz et al. 2011], e possuem aplicações em áreas como simplificação textual [Aluisio et al. 2008], tradução automática [Aziz et al. 2008, Aziz et al. 2009] e outras.

Na geração a partir de Árvore Subespecificada, o sistema recebe como entrada uma árvore da sentença a ser gerada e invoca de forma autônoma todos os métodos que se façam necessários para impor concordância e linearização da sentença. Esta especificação pode ser incompleta, e o resultado obtido é assim apenas uma aproximação baseada em regras de linearização simples, o que pode nem sempre corresponder à ordem ideal dos termos. Para geração de sentenças mais complexas com total controle sobre o resultado, faz-se necessário o uso do sistema na modalidade de geração Incremental (seção 2.1).

A árvore de entrada contém os nós *agent*, *action* e, opcionalmente, *patient*, nesta ordem. Os termos *agent* e *patient* são conceitos a serem realizados como SNs, enquanto *action* representa um SV. Na prática, entretanto, SNs e SVs são apenas conjuntos de palavras a serem ordenadas pelo sistema. Um exemplo é apresentado a seguir.

(1) O escritor indiano Amitav Ghosh
concept(s15,obj, [s, ['indiano'], head('escritor'), ['amitav-ghosh'], def, m]).

Exceto pela declaração do núcleo, todos os demais termos são opcionais e, se necessário, serão inferidos com base em valores *default* e/ou traços morfológicos dos outros termos do sintagma. Por simplicidade, assume-se que sintagmas preposicionais inseridos na expressão principal com uso de preposições já estejam em sua forma final.

O exemplo (2) a seguir ilustra uma definição de conceito representado por um SN contendo um termo aninhado, e o exemplo (3) especifica um SV. A especificação de uma sentença consiste assim em referenciar os sintagmas s15, v15 e p15 na ordem desejada (4):

(2) festa de aniversário
concept(p15,obj, [f, s, rel(['de', 'aniversário']), head('festa')]).

(3) provavelmente não terá
concept(v15, act, [ind, ['provavelmente'],fut, head(['ter']), vfin], ['não']).

(4) O escritor indiano Amitav Ghosh provavelmente não terá festa de aniversário
sentence(u15, [agent(s15), action(v15), patient(p15)]).

3. Avaliação

A presente discussão será limitada a uma avaliação do sistema na modalidade de geração a partir de árvore subespecificada descrito na seção 2.2, em comparação com o uso de modelos estatísticos simples [Pereira and Paraboni 2008]. Uma avaliação baseada em modelos mais sofisticados é apresentada em [de Novais and Paraboni 2012], na qual o presente trabalho atua como um dos sistemas de *baseline* considerados.

Consideramos nesta avaliação três sistemas de *baseline* estatísticos de 2, 3 e 4-gramas [Pereira and Paraboni 2007] construídos segundo a técnica de geração em 2 estágios proposta em [Langkilde 2000]. Estes sistemas utilizam modelos estatísticos de língua treinados a partir de uma versão expandida do corpus NILC [Nunes et al. 1996] totalizando 141 milhões de palavras.

Como conjunto de teste foi utilizado o corpus de teste utilizado em [de Novais and Paraboni 2012], contendo 4.297 manchetes de jornal on-line do ano de 2009 considerando-se uma porção distinta da utilizada como corpus de treinamento. Estas sentenças sofreram um processo de abstração semelhante ao adotado na preparação dos dados adotado no *Generation Challenges 2011 Surface Realisation Shared Task* [Belz et al. 2011], resultando na representação discutida na seção 2 deste trabalho.

Cada uma das 4.297 entradas foi fornecida ao sistema proposto e aos três sistemas de *baseline*, comparando-se o *string* produzido e a frase de referência do corpus de teste. Os resultados obtidos apontam uma vantagem do sistema proposto, com acurácia de 69%, sobre os demais sistemas, que obtiveram acurácia de 63% (modelo de 2-gramas) e 66% (3-gramas e 4-gramas). Esta diferença é significativa segundo o teste de chi-quadrada sob os valores de acurácia ($\chi^2 = 8,62, df = 1, p < 0,0033$).

4. Considerações

Este artigo apresentou um sistema de realização superficial baseado em regras para o Português brasileiro. Os resultados superam os obtidos com uso de modelos de n-gramas padrão, sugerindo uma solução de baixo custo para o problema, aos moldes do sistema existente para a língua inglesa [Gatt and Reiter 2009]. Por outro lado, os resultados apresentados ainda são inferiores aos observados em estudos mais recentes com modelos estatísticos ditos fatorados, ou FLMs [Bilmes and Kirchhoff 2003] utilizados em [de Novais and Paraboni 2012].

Sistemas estatísticos, entretanto, possuem um custo computacional elevado. O uso de FLMs como discutido em [de Novais and Paraboni 2012], por exemplo, pode não ser viável em aplicações de geração de língua natural em tempo real. A abordagem baseada em regras, por outro lado, não sofre destas limitações.

Como trabalho futuro, o presente protótipo será documentado e expandido na forma de um sistema completo para uso de pesquisadores interessada no processamento da língua portuguesa.

Agradecimentos

Este trabalho contou com apoio FAPESP (2009/08499-9 e 2009/09061-7), e do programa Ensinar com Pesquisa da Universidade de São Paulo.

References

Aluisio, S. M., Specia, L., Pardo, T. A. S., Maziero, E., and Fortes, R. P. M. (2008). Towards brazilian portuguese automatic text simplification systems. In *The ACM Symposium on Document Engineering*, pages 240–248.

Aziz, W. F., Pardo, T. A. S., and Paraboni, I. (2008). An experiment in spanish-portuguese statistical machine translation. In *SBIA-2008*, pages 248–257.

Aziz, W. F., Pardo, T. A. S., and Paraboni, I. (2009). Statistical phrase-based machine translation: Experiments with brazilian portuguese. In *ENIA-2009*, pages 769–778.

Belz, A., White, M., Espinosa, D., Kow, E., Hogan, D., and Stent, A. (2011). The first surface realisation shared task: Overview and evaluation results. In *13th European Workshop on Natural Language Generation (ENLG)*, pages 217–226.

Bilmes, J. and Kirchhoff, K. (2003). Factored language models and generalized parallel backoff. In *HLT-NAACL 2003 conference*.

Cuevas, R. and Paraboni, I. (2008). A machine learning approach to portuguese pronoun resolution. *LNAI*, 5290:262–271.

de Lucena, D. J., Pereira, D. B., and Paraboni, I. (2010). From semantic properties to surface text: The generation of domain object descriptions. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, (45):48–58.

de Novais, E. M. and Paraboni, I. (2012). Portuguese text generation using factored language models. *Journal of the Brazilian Computer Society*, 19:135–146.

Gatt, A. and Reiter, E. (2009). SimpleNLG: A realization engine for practical applications. In *European Natural Language Generation workshop (ENLG-2009)*.

Langkilde, I. (2000). Forest-based statistical sentence generation. In *Proceedings of ANLP-NAACL'00*, pages 170–177.

Malouf, R. (2000). The order of prenominal adjectives in natural language generation. In *38th Meeting of the ACL*.

Mitchell, M. (2009). Class-based ordering of prenominal modifiers. In *12th European Workshop on Natural Language Generation*, pages 50–57.

Nunes, M. G. V., Vieira, F. M. C., Zavaglia, C., Sossolote, C. R. C., and Hernandez, J. (1996). A construção de um léxico para o português do Brasil: lições aprendidas e perspectivas. In *II PROPOR*, pages 61–70.

Paraboni, I. (2003). *Generating references in hierarchical domains: the case of Document Deixis*. PhD thesis.

Paraboni, I., Masthoff, J., and van Deemter, K. (2006). Overspecified reference in hierarchical domains: measuring the benefits for readers. In *INLG-2006*, pages 55–62.

Pereira, D. B. and Paraboni, I. (2007). A language modelling tool for statistical NLP. In *TIL-2007*, pages 1679–1688.

Pereira, D. B. and Paraboni, I. (2008). Statistical surface realisation of portuguese referring expressions. In *LNAI*, volume 5221, pages 383–392. Springer-Verlag.

Reiter, E. (2007). An architecture for data-to-text systems. In *ENLG-2007*, pages 97–104.