Um Sistema de Realização Superficial para Geração de Textos em Português

Douglas Fernandes Pereira da Silva Junior [1]
Ivandré Paraboni [1]
Eder Miranda de Novais [1]

Resumo: Sistemas de geração de língua natural (GLN) - que produzem texto a partir de dados não-linguísticos - possuem uma ampla gama de aplicações em visualização textual de conteúdos complexos e/ou em grandes volumes. Este trabalho enfoca a implementação de um módulo de realização textual baseado em regras para o português brasileiro, chamado *PortNLG*, que trata da tarefa de linearização sentencial para aplicações computacionais que necessitem apresentar dados de saída em formato textual. *PortNLG* é apresentado na forma de uma biblioteca JAVA, e seus resultados são superiores aos de modelos de n-gramas na tarefa de geração de manchetes de jornal.

Abstract: Natural Language Generation (NLG) systems - which produce text from non-linguistic data - are employed in a variety of applications for visualizing complex or otherwise large data sets. This paper focuses on the implementation of a rule-based surface realisation component for the Brazilian Portuguese language, called *PortNLG*, which addresses the task of sentence linearisation for applications that are required to produce output text. *PortNLG* is presented as a JAVA library, and it outperforms the use of standard n-gram models applied to the generation of newspapers headlines.

1 Introdução

A visualização de conteúdos de bases de dados complexos e/ou em grandes volumes é frequentemente feita com uso de documentos de textos em língua natural. Sistemas de Geração de Língua Natural (GLN) - que produzem descrições textuais a partir de uma entrada de dados não-linguística [1] - são assim empregados quando texto predefinido não é suficiente, ou seja, quando é necessária uma maior variação linguística nos documentos gerados e/ou maior proximidade em relação ao desempenho humano. Aplicações típicas de GLN incluem, por exemplo, a geração de relatórios descritivos do mercado de ações a partir de indicadores financeiros [1], boletins de previsão do tempo gerados a partir de dados de satélites [2], diagnósticos médicos produzidos a partir da leitura de sensores de equipamentos hospitalares [3] e muitas outras.

[1]Escola de Artes, Ciências e Humanidades (EACH), Universidade de São Paulo (USP). Av. Arlindo Bettio, 1000 - São Paulo, Brasil
{douglas.fernandes.silva, ivandre, eder.novais @ usp.br}

Tendo como ponto de partida um objetivo de comunicação de alto nível, um sistema de GLN constrói gradativamente um plano para representar este objetivo em forma textual. Ao longo do processo de geração, o plano inicial sofre uma série de transformações, cada qual constituindo uma forma de representação intermediária do conhecimento comunicado, até o ponto em que o texto em língua natural é produzido. A arquitetura de um sistema de GLN pode assim ser vista como um *pipeline* de três estágios [1] dividida em módulos de *planejamento do documento* [4], *planejamento de sentenças* [5, 6, 7] e *realização superficial* [8, 9, 10, 11].

Este trabalho enfoca a implementação e avaliação de um protótipo de sistema de realização superficial para o português brasileiro. Mais especificamente, tratamos da última etapa da representação do conhecimento em uma língua-alvo (i.e., em português), pressupondo que o conteúdo a ser realizado já foi definido, e que portanto as tarefas de escolha lexical e seleção da estrutura sentencial (e.g., em voz ativa, passiva etc.) já foram efetuadas.

O sistema proposto - chamado *PortNLG* - é inteiramente baseado em regras, e pretende-se neste artigo demonstrar que este paradigma pode ser superior ao uso de modelos estatísticos de língua baseados em n-gramas, que são amplamente empregados nesta tarefa [12, 13, 14, 15, 16]. Assim como [11], *PortNLG* 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.

O restante deste artigo está organizado da seguinte forma. Na seção 2 são discutidos trabalhos relacionados à presente proposta. Na seção 3 é descrito o sistema *PortNLG* propriamente dito, ilustrando-se suas duas formas de utilização (aqui chamadas de geração Incremental e geração a partir de Árvore subespecificada) na forma de exemplos práticos. Na seção 4 é apresentado o trabalho de avaliação do sistema, que é comparado a uma série de geradores estatísticos e outros sistemas de *baseline*. Finalmente, a seção 5 apresenta conclusões e indicações de trabalho futuro.

2 Contextualização

2.1 A tarefa computacional de realização superficial

A realização superficial é essencialmente 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 [1]. A entrada de um módulo de realização textual é algum tipo de especificação abstrata da sentença, e a saída é uma cadeia de caracteres em língua natural.

A especificação de entrada é uma questão fundamental a ser considerada no projeto

de um realizador superficial: evidentemente, quanto mais próxima esta representação for da forma superficial, menor será o esforço de realização. Ao simplificarmos a tarefa desta forma, entretanto, as decisões de realização estão apenas sendo transferidas para a etapa anterior do *pipeline* de GLN, isso é, o planejamento sentencial.

Não há de fato um limite claro entre o planejamento sentencial e realização textual, e a decisão sobre o que esperar como entrada para a realização textual é ditada pela aplicação subjacente e o grau de especificidade semântica que esta pode fornecer [1]. Por exemplo, se a aplicação for capaz de fornecer instruções já detalhadas sobre a lexicalização [17, 18] e a estrutura sintática desejada, então a tarefa de realização textual consistirá basicamente em aplicar um conjunto de regras gramaticais adequado para impor concordância, linearização dos termos da sentença etc.

Todos sistemas de GLN incluem algum tipo de módulo de realização superficial em sua arquitetura. Apenas recentemente, entretanto, começam a surgir iniciativas de separação desta tarefa de outros aspectos da geração de língua natural. De especial interesse para a presente discussão é a divisão da tarefa em dois componentes relativamente independentes, tal qual proposto em [11]: um componente tático, mais dependente da aplicação, encarregado de mapear conceitos de domínio para algum tipo de estrutura abstrata de motivação linguística, e um componente operacional encarregado de impor restrições gramaticais e demais operações dependentes da língua-alvo. O presente trabalho ocupa-se unicamente do componente operacional de GLN.

Além das dificuldades de definição da especificação de entrada, ou talvez por causa destas, a realização superficial operacional ainda possui poucos representantes evidentes na literatura da área. Além disso, por tratar-se de uma tarefa computacional altamente dependente da língua-alvo (no presente caso, o português), a comparação com sistemas existentes para idiomas como o inglês é na prática pouco útil, já que os desafios apresentados são diferentes. A discussão na próxima seção é assim limitada a sistemas recentes de língua inglesa que se aproximam dos objetivos da presente proposta, sem no entanto tratar exatamente do mesmo problema linguístico.

2.2 Trabalho Relacionado

Uma recente tentativa de padronização da tarefa de realização superficial operacional foi o primeiro *Surface Realisation Shared Task* [19][2], evento dedicado à avaliação conjunta de realizadores para a língua inglesa baseados em uma mesma entrada de dados abstraída do *Wall Street Journal corpus*[3].

O desafio proposto em [19] foi completado por cinco equipes participantes, sendo três

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

[3] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2000T43

sistemas estatísticos e dois simbólicos. Os resultados apresentados em [20] sugerem que os sistemas estatísticos tiveram desempenho médio superior aos simbólicos.

Apesar do esforço de padronização da tarefa enunciada em [19] para manter a clareza e objetividade na comparação entre os sistemas participantes, em [20] foram levantadas diversas questões relacionadas à especificação de entrada e o tipo de conhecimento externo que cada sistema poderia utilizar. Estas questões ilustram a dificuldade em estabelecer de forma precisa o escopo da tarefa de realização superficial em diferentes aplicações.

Dos cinco sistemas participantes do *Shared task* apresentados em [20], dois eram baseados em conhecimento profundo, e assim fogem do escopo do presente trabalho. Os outros três sistemas são no entanto de maior interesse para a presente proposta, pois tratam da mesma tarefa de organização de sentença que pretendemos abordar, embora obviamente baseados em uma representação linguística distinta (em inglês) e gerando texto também neste idioma.

O trabalho apresentado em [21] consiste de um realizador superficial estatístico que faz uso de modelos de n-gramas baseados em dependências sintáticas. Em [21] estes modelos são usados para tirar proveito da informação estrutural da sentença e de diversos outros atributos linguísticos, e assim restringir o tamanho do espaço de geração.

O trabalho em [22] também segue uma abordagem estatística, extraindo toda informação relevante para a tarefa diretamente de um corpus - o que sob certo ponto de vista acrescenta uma quantidade de conhecimento externo considerável ao sistema propriamente dito. Estas informações incluem a geração de um modelo de árvores locais, um dicionário morfológico e um modelo estatístico de língua baseado em trigramas.

O terceiro sistema de interesse na competição, discutido em [23], é o único a seguir uma abordagem estatística não-superficial. Este sistema faz uso de gramáticas da língua inglesa baseadas em operações de unificação, ou *unification-based grammars*.

Apesar da similaridade de objetivos com *PortNLG*, entretanto, reiteramos que a diferença em língua-alvo (i.e., inglês versus português) torna uma comparação direta com o presente trabalho inviável. Assim, e de forma mais diretamente relevante para a presente discussão, *PortNLG* pode ser visto como um equivalente para língua portuguesa do sistema *SimpleNLG* [11] para o inglês.

O sistema descrito em [11] é um realizador superficial para o inglês baseado em regras gramaticais daquele idioma. Este sistema é disponibilizado na forma de uma biblioteca de métodos JAVA para construção rápida (ou simples, conforme o nome do sistema sugere) de sintagmas e sentenças completas. Entretanto, *PortNLG* não é uma mera versão em português do sistema em [11]. Detalhes são fornecidos na próxima seção.

3 O Sistema *PortNLG*

Assim como em [11], *PortNLG* 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 e desenvolvimento orientado a objetos para sua utilização efetiva. Feitas estas considerações, empregamos aqui o termo 'sistema' em seu sentido amplo, e que não deve ser confundido com um sistema completo de GLN. O realizador superficial aqui discutido não inclui, por exemplo, os demais módulos deste tipo de sistema [1], e não possuir interface com usuário.

Em virtude de certas peculiaridades da língua portuguesa, entretanto, o presente trabalho guarda uma série de diferenças em relação ao *SimpleNLG*. Dentre estas, a mais significativa é a arquitetura do sistema, que no presente caso se tornou mais complexa pela necessidade de uso da base lexical implementada em [24] para obtenção de flexões de gênero, número e outras. No caso do sistema apresentado em [11], por outro lado, o sistema utiliza apenas regras simples para transformação de palavras sem uso de recursos lexicais adicionais, já que a língua inglesa possui um grau muito menor de variação.

A figura 1 ilustra a organização geral do sistema *PortNLG*. A entrada do sistema é uma especificação abstrata da sentença a ser gerada conforme discutido a seguir. Fazendo uso da base lexical implementada em [24] e regras gramaticais próprias, o sistema constrói a cadeia correspondente em português, a qual constitui o resultado final do processo.

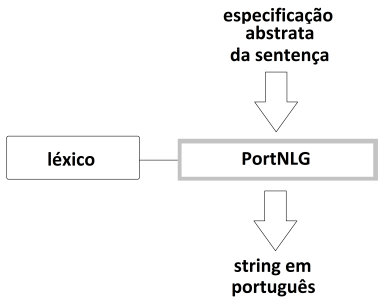

Figura 1. Visão geral do sistema *PortNLG*

Uma diferença importante entre o presente trabalho e o sistema em [11] é a forma de chamada ao sistema e a especificação de entrada. Como forma de tornar o presente sistema mais facilmente adaptável a possíveis aplicações, foram implementadas duas modalidades de chamada ao sistema: uma modalidade, ao estilo utilizado em [11], em que a aplicação sub-

jacente realiza uma série de chamadas incrementais aos métodos de construção da sentença, e uma modalidade de chamada única, na qual é fornecida uma árvore representando a sentença a ser gerada, semelhante ao padrão de entrada dito 'superficial' adotado na competição *Surface Realisation Shared Task* [19].

As duas modalidades de chamada ao sistema fazem uso dos mesmos métodos básicos de construção da sentença, porém implementando funcionalidades ligeiramente distintas, e tendo também motivações diferentes. Em ambos os casos, entretanto, a realização superficial é desempenhada de forma totalmente automática, ou seja, a única diferença entre elas está no formato dos dados fornecidos como entrada. Nota-se também que usamos o termo "sistema" de forma ampla aqui, mas que o trabalho desenvolvido é na verdade um módulo de realização superficial a ser acoplado a uma aplicação subjacente.

A construção de uma frase mediante chamadas incrementais é a maneira tradicional de uso do sistema, na qual a aplicação possui total controle sobre a forma superficial produzida. Assim como em [11], a geração de uma sentença neste caso requer a escrita de um pequeno programa JAVA que invoca os métodos construtores desejados. Exemplos destas chamadas são apresentados na seção 3.1.

Na construção da sentença mediante chamada única, por outro lado, não há nenhum esforço de programação. Neste caso, o sistema espera receber como entrada uma representação simbólica - em formato de árvore - da sentença a ser gerada. Esta árvore não necessariamente precisa conter todas as informações necessárias para a produção da sentença, ou seja, pode ser subespecificada, e neste caso o sistema pode ter que completar as informações necessárias para geração da sentença (em especial, para a tarefa de linearização) de forma autônoma. Esta modalidade de geração é discutida na seção 3.2.

3.1 Geração Incremental

Em sua forma mais típica de utilização, *PortNLG* é invocado mediante uma série de chamadas a métodos de construção de sintagmas e outros componentes da sentença[4]. Os métodos disponibilizados pelo sistema são divididos em cinco categorias: sintagma nominal, verbal, preposicional, adverbial e adjetival. A criação de uma sentença completa consiste em invocar estes métodos conforme necessário 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.)

No restante desta seção discutimos a construção dos principais componentes de uma sentença utilizando exemplos práticos de uso da biblioteca. Informações detalhadas são apresentadas na documentação do sistema mediante solicitação aos autores.

[4]Cabe destacar entretanto que o sistema não objetiva seguir uma teoria linguística específica, sendo orientado para aplicações computacionais práticas nas quais faz-se necessário gerar texto.

A classe *SintagmaNominal* permite a criação de sete tipos de sintagma nominal (SN): nome próprio (prop), pronome pessoal (prop_pers), determinante possessivo (poss), descrição definida (def), indefinida (indef), determinante quantificador (quant) ou numeral (num). A criação de um SN começa assim invocando-se seu método construtor, como nos exemplos a seguir:

SintagmaNominal sn_1 = new poss();
SintagmaNominal sn_2 = new num();
SintagmaNominal sn_3 = new def();

Uma vez que o SN seja criado, deve-se especificar seus parâmetros *Lemma* (i.e., a forma-base do núcleo do SN) e *DetLemma* (o núcleo do termo determinante). No exemplo abaixo, é criado um SN com determinante do tipo possessivo e núcleos indicados:

SintagmaNominal sn = SintagmaNominal.poss();
sn.setLemma("urso");
sn.setDetLemma("eu");

A representação textual propriamente dita é obtida pelo método *getRepresentacaoTextual*, resultando em um SN com determinante possessivo na forma 'meu urso':

sn.getRepresentacaoTextual()

Opcionalmente, o SN pode incluir modificadores diversos, incluindo outros sintagmas. É possível especificar que o modificador ocupe uma posição anterior ou posterior ao núcleo, ou deixar que o sistema tente escolher a posição mais adequada. Por exemplo, é possível acrescentar a informação de quantidade ao SN do exemplo anterior, resultando na cadeia 'meus dois ursos':

sn.setModificadorNumerais("dois", Ordem.Padrao());

Um SN pode ter seus parâmetros gênero, número, grau, caso e pessoa modificados como segue, no qual o método Null() indica que a informação em questão não é relevante para a construção daquele tipo de SN. O exemplo a seguir ilustra a geração da cadeia 'minhas duas ursas':

sn.setParametrosInt(Genero.Feminino(), Numero.Plural(),Grau.Null(),Caso.Null(), Tempo.Null());

Sintagmas adjetivais (classe *SintagmaA*) são cadeias de adjetivos ligados por conjunções, criados primariamente com o propósito de servir de modificadores para SNs. Da

mesma forma que no caso dos SNs, também no caso dos sintagmas adjetivais é possível modificar parâmetros como gênero e número, bem como a ordem linear dos componentes. Um exemplo completo de criação de sintagma nominal contendo dois modificadores do tipo *SintagmaA* é apresentado a seguir, resultando na cadeia 'as minhas lindas ursinhas carinhosas':

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

SintagmaA sa_2 = new SintagmaA();
sa_2.setAdjetivo("carinhoso");

SintagmaNominal sn = SintagmaN.def();
sn.setLemma("urso");
sn.setDetLemma("o");
sn.setModificadorPronomePoss("eu", Ordem.antes());
sn.setModificadorSintagmaAdjetivo(sa_1, Ordem.antes());
sn.setModificadorSintagmaAdjetivo(sa_2, Ordem.depois());
sn.setParametrosInt(Genero.Feminino(), Numero.Plural(),Grau.Diminutivo(),Caso.Null(), Tempo.Null());

Sintagmas preposicionais (classe *SintagmaP*) são compostos de uma preposição seguida de um SN chamado de termo consequente, e podem ser anexados a um SN já existente com uso do método *SetPPlist* da classe *SintagmaNominal*. A seguir apresentamos um exemplo completo resultando na cadeia 'um carro de brinquedo'.

SintagmaNominal sn_1 = SintagmaN.none();
sn_1.setLemma("brinquedo");
SintagmaP sp = new SintagmaP();
sp.setPreposicao("de");
sp.setConsequente(sn_1);

SintagmaNominal sn = SintagmaN.indef();
sn.setLemma("carro");
sn.setDetLemma("um");
sn.SetPPlist(sp);

A classe *SintagmaVerbal* permite a criação de sintagmas verbais (SV) de cinco tipos: finito simples (vfin), finito composto (vfin2), infinitivo (vinf), gerúndio (vger) e particípio (vpcp), e seus parâmetros gênero, número, pessoa, modo e tempo podem ser modificados pelos métodos correspondentes como nos casos anteriores. A seguir é apresentado um exemplo completo de construção do SV 'estou dormindo'.

SintagmaVerbal sv = SintagmaV.VGer();
sv.setVerboPrincipal("estar");
sv.setVerboSegundario("dormir");
sv.setParametroInt(Genero.Null(),Numero.Null(), Pessoa.Null(), Modo.Null(), Tempo.Presente());

Sintagmas adverbiais (classe *SintagmaAdverbial*) são criados como cadeias de advérbios e termos auxiliares. A seguir um exemplo de criação de um sintagma deste tipo usado como modificador para o SV criado no exemplo anterior, no qual é especificada também a sua posição em relação ao núcleo do SV resultante 'estou dormindo moderadamente'.

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

Finalmente, SNs e SVs com modificadores opcionais podem ser combinados formando uma sentença completa que pode assumir quatro formas (ou templates) pré-definidos: voz ativa, voz passiva, imperativo e sujeito oculto. Um exemplo completo de geração da sentença em voz passiva 'o carro de brinquedo foi comprado pelo homem' é apresentado a seguir.

SintagmaVPcp sv = new SintagmaVPcp();
sv.setVerboPrincipal("ir");
sv.setVerboSegundario("comprar");

SintagmaNominal sn_1 = new none();
sn_1.setLemma("homem");

SintagmaNominal sn_2 = new none();
sn_2.setLemma("brinquedo");
SintagmaP sp = new SintagmaP();
sp.setPreposicao("de");
sp.setConsequente(sn_2);

SintagmaNominal sn_3 = new indef();
sn_3.setLemma("carro");
sn_3.setDetLemma("o");
sn_3.SetPPlist(sp);

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

3.2 Geração a partir de Árvore Subespecificada

Embora a especificação de entrada completa descrita na seção anterior seja a forma padrão de gerar textos a partir de aplicações que lidam com dados não-linguísticos, há casos em que a informação a ser gerada já se encontra parcialmente disponível em forma textual. Problemas de realização superficial desta natureza, chamados de geração Texto-para-Texto, têm sido abordados, por exemplo, na recente competição de sistemas de realização superficial

(*Surface Realisation Shared Task*, em [19]), e possuem diversas aplicações em áreas como simplificação textual [25], tradução automática [26, 27] e outras. Assim, como forma de complementar o sistema proposto, e também como forma de viabilizar a avaliação automática do mesmo com base em corpus conforme descrito na próxima seção, optamos por implementar uma funcionalidade adicional - chamada de geração a partir de Árvore Subespecificada - capaz de lidar também com este tipo de entrada. Na versão atual do sistema, entretanto, há suporte apenas à geração de sentenças em voz ativa.

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 (descritos na seção anterior) que se façam necessários para impor concordância e linearização da sentença. Cabe destacar no entanto que, uma vez que esta especificação de entrada pode não conter todas as informações necessárias para linearização de alguns termos (como por exemplo componentes de sintagmas nominais), o resultado obtido é apenas uma aproximação baseada em regras de linearização simples, o que em casos mais complexos pode não 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 3.1) e fornecendo-se uma especificação de entrada completa.

A árvore representando a sentença de entrada em voz ativa contem os nós *agent*, *action* e, opcionalmente, *patient*, sempre nesta ordem fixa. Os termos *agent* e *patient* são conceitos a serem realizados como sintagmas nominais, enquanto que o termo *action* representa um sintagma verbal. Na prática, entretanto, sintagmas nominais e verbais são apenas conjuntos (não ordenados) de palavras de conteúdo, cabendo ao sistema impor a ordenação correta a cada um. Assim, sintagmas nominais são conjuntos de substantivos, nomes próprios, adjetivos e sintagmas preposicionais, enquanto que sintagmas verbais são conjuntos de verbos, preposições e advérbios. Em ambos os casos a estrutura inclui a especificação de um termo núcleo (*head*) do sintagma representando o seu conceito central (nome ou verbo principal) ao qual os demais termos estão subordinados. A seguir é apresentado um exemplo de sintagma nominal e sua especificação segundo este formato, e cujo núcleo foi definido como 'cantor'.

(1) a cantora americana christina aguilera
concept(s15,obj, [s, ['americano'], head('cantor'), ['christina=aguilera'], def, f]).

Nesta especificação, observa-se que o conceito é representado por uma lista não ordenada de palavras de conteúdo, com indicação do núcleo do sintagma e informações adicionais sobre o tipo de determinante a ser empregado (*def* corresponde a uma descrição definida etc.) e informações de gênero e número referentes ao núcleo, e que no caso do sintagma nominal serão impostas aos demais componentes da estrutura.

Exceto pela declaração do núcleo, todos os demais termos são opcionais. Por exemplo, na falta de uma instrução sobre o tipo de determinante a empregar, o sistema gera uma

descrição definida. Se as informações de gênero ou número forem omitidas, o sistema terá de inferi-las com base nos traços morfológicos dos outros termos do sintagma. Nota-se assim que no exemplo (1) acima não seria necessário especificar o gênero (f), já que um de seus componentes é o nome próprio 'Christina Aguilera', que consta na base lexical do sistema apenas na forma feminina, e impõe-se a todos os componentes que tenham flexão de gênero (em especial, o termo 'cantor' será corretamente realizado como 'cantora').

Por razões de eficiência, esta especificação de entrada assume 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 sintagma nominal contendo um termo aninhado deste tipo.

(2) programa de rádio
concept(p15,obj, [m, s, rel(['de', 'rádio']), head('programa')]).

Expressões como 'programa de rádio' acima possuem assim um núcleo a ser transformado em cadeia de caracteres (efetuando-se flexão de gênero e número, que no exemplo acima resultaria na mesma palavra 'programa') e um subcomponente já realizado ('de rádio') representados por cláusulas do tipo *rel* (relação). Cabe observar entretanto que estes subcomponentes poderiam também ser gerados pelo próprio sistema de forma recursiva.

Finalmente, um exemplo de especificação de sintagma verbal é apresentado a seguir. Sintagmas verbais podem incluir advérbios e preposições. Caso o tempo ou modo verbal não sejam especificados, é utilizado por *default* o pretérito simples do modo indicativo.

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

Com base nestas definições, o trabalho do sistema consiste em interpretar a especificação em árvore e invocar os métodos necessários para construir cada um dos componentes da sentença. Um exemplo de especificação de sentença em voz ativa (agente - ação - paciente) é apresentado a seguir. Os identificadores s15, v15 e p15 referem-se aos conceitos definidos nos exemplos (1-3) acima, que no caso do template de voz ativa são tratados como agente, ação e paciente da sentença, respectivamente[5]:

(4) a cantora americana christina aguilera não terá programa de rádio
sentence(u15, [agent(s15), action(v15), patient(p15)]).

[5]Na prática, esta especificação segue a sintaxe de cláusulas PROLOG, linguagem utilizada no desenvolvimento deste módulo do sistema.

4 Avaliação

O presente sistema foi recentemente empregado como *baseline* em um amplo exercício de avaliação de outro projeto na área, descrito em [28]. A presente discussão será assim limitada a uma avaliação do sistema na modalidade de geração a partir de árvore subespecificada descrito na seção 3.2, objetivando demonstrar que a presente abordagem pode ser superior ao uso de modelos estatísticos de língua [12, 13, 14, 15, 16].

Na avaliação de um sistema de realização textual coloca-se a questão de como identificar possíveis oportunidades de melhoria. Na modalidade de execução original do sistema (i.e., com chamadas sucessivas aos métodos de construção de sintagmas descrita na seção 3.1) estas oportunidades são pouco evidentes, já que dada uma especificação de entrada completa o sistema simplesmente implementa aquilo que é solicitado. Além disso, como a geração de uma frase exige uma série de chamadas sucessivas aos métodos disponibilizados, testar a geração de um grande número delas de forma manual teria um custo operacional elevado.

Consideramos assim que a forma mais relevante de avaliar as funcionalidades de geração do sistema é utilizando-se o modo de geração a partir de árvore subespecificada descrito na seção 3.2. Além de nos permitir avaliar o desempenho de todos os métodos em condições de entrada de dados incompleta (e sujeitas a uma certa margem de erro), este tipo de especificação pode ser facilmente obtido a partir de um corpus de sentenças reais que, a exemplo dos sistemas participantes da competição detalhada em [19], proporciona uma oportunidade de avaliação automática em grande escala do sistema.

Para fins de avaliação foram utilizados cinco sistemas de *baseline*. Três destes são sistemas estatísticos simples baseados em n-gramas [29] de ordem 2, 3 e 4 semelhantes aos discutidos em [30], construídos segundo a técnica de geração em 2 estágios proposta em [12]. Resumidamente, estes sistemas utilizam modelos estatísticos de língua treinados a partir de uma versão expandida do corpus NILC [31], originalmente com cerca de 32 milhões de palavras, ao qual foram adicionados os artigos on-line do período de janeiro de 2006 a fevereiro de 2011 da Folha de São Paulo[6] e da revista Veja[7]. Após uma série de tarefas de pré-processamento para remoção de ruído (e.g., trechos de artigos em outros idiomas etc.), duplicidades e erros diversos, foi obtido um corpus de treinamento de cerca de 141 milhões de palavras no total.

O corpus de treinamento foi etiquetado com a ferramenta MXPOST disponibilizada pelo projeto LACIO-WEB[8] e teve parte de suas etiquetas corrigidas de forma semiautomática com base em uma amostragem de erros frequentes (em especial, para correta identificação de nomes próprios). O corpus foi então empregado no treinamento de modelos de língua de

[6] www.folha.com.br
[7] www.veja.com.br
[8] http://nilc.icmc.sc.usp.br/nilc/tools/nilctaggers.html

ordem 2, 3 e 4 correspondendo ao núcleo dos três métodos de *baseline* estatísticos considerados. Em todos os casos, dada uma especificação de entrada, a tarefa do sistema consiste em produzir todas as permutações possíveis de saída. Estas saídas são então submetidas ao filtro estatístico, e a permutação mais provável é selecionada como a saída do sistema correspondente e as demais alternativas são simplesmente descartadas. Detalhes sobre esta técnica são apresentados em [30].

Os mesmos conjuntos de permutações produzidos para geração estatística foram também empregados na avaliação de um sistemas de *baseline* aleatório, que simplesmente seleciona uma permutação qualquer como sentença de saída. Finalmente, foram ainda considerados nesta análise os resultados do sistema baseado em template de ordem fixa discutido em [28]. Neste sistema, campos de um template da sentença são preenchidos da esquerda para a direita na mesma ordem dos dados fornecidos como entrada.

Para execução de um teste de ampla cobertura com base em exemplos reais de uso da língua, foi utilizado o corpus de teste 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 *Surface Realisation Shared Task* [19], no qual cada palavra de conteúdo foi substituída pela sua forma-base, e a estrutura da sentença foi representada na forma de árvores de dependências com ordem linear aleatória. O exemplo de especificação de entrada completa ilustrado na seção 2 para a sentença-alvo 'a cantora americana christina aguilera não terá programa de rádio' é reproduzido abaixo:

```
sentence(u15, [agent(s15), action(v15), patient(p15) ]).
concept(s15,obj, [s, ['americano'], head('cantor'), ['christina=aguilera'], def, f ]).
concept(v15, act, [ind, ['ainda'],fut, head(['ter']), vfin], ['não']).
concept(p15,obj, [m, s, rel(['de', 'rádio' ]), head('programa') ]).
```

Cada uma das 4.297 entradas foi fornecida ao sistema conforme proposto na seção anterior e aos cinco sistemas de *baseline* (sendo três estatísticos e dois não-estatísticos). A avaliação propriamente dita consistiu da comparação entre a frase produzida e a frase de referência do corpus de teste, utilizando-se as métricas de distância de edição (i.e., número de operações de inserção, exclusão e substituição necessárias para igualar a cadeia do sistema e a cadeia do conjunto de referência), exatidão (i.e., contagem de coincidência total entre cadeias de caracteres, NIST [32] e BLEU [33]. Os resultados obtido com cada uma destas métricas de avaliação são apresentados na Tabela 1.

Tabela 1. Resultados da avaliação do sistema

System	Exatidão	Distância	BLEU	NIST
Aleatório	0,22	15,52	0,55	13,99
Template fixo	0,62	5,52	0,81	14,58
2-gramas	0,63	6,40	0,82	14,67
3-gramas	0,66	5,80	0,83	14,72
4-gramas	0,66	5,79	0,83	14,72
PortNLG	0,69	4,55	0,85	14,97

Resultados de análise de variância de fator único (ANOVA) sobre os valores de distância de edição seguida de teste de Tukey HSD ($\alpha = 0,05$) indicam diferença significativas entre os sistemas (F(5,25776)=869,64, MSE=82,67, p<0,0001). Os subconjuntos homogêneos obtidos são ilustrados na Tabela 2.

Sistema				
PortNLG	A			
Template fixo		B		
4-gramas		B		
3-gramas		B		
2-gramas			C	
Aleatório				D

Tabela 2. Subconjuntos homogêneos para distância de edição. Sistemas que não possuem uma letra em comum são significativamente distintos para $\alpha = 0,05$.

Devido ao tamanho relativamente pequeno das sentenças de teste (até 9 palavras), observa-se que o uso de n-gramas de ordem 4 pelo sistema 4W não melhora o desempenho em relação ao uso de n-gramas de ordem 3 (sistema 3W). Esta diferença, entretanto, pode se tornar significativa à medida que sentenças mais longas venham a ser consideradas.

Os resultados obtidos apontam também uma vantagem do sistema proposto sobre os demais tomando-se por base os coeficientes de exatidão. A diferença entre *PortNLG* e os sistemas de segundo melhor resultado em termos de exatidão (i.e., os modelos $4w/3w$) é significativa segundo o teste de chi-quadrada ($\chi^2 = 7,65, df = 1, p = 0,05$). Além disso, os modelos $3w/4w$ são significativamente superiores ao modelo $2w$ e de template fixo ($\chi^2 =$

$10,54, df = 1, p < 0,001$), que por sua vez são significativamente superiores à abordagem aleatória ($\chi^2 = 1484,98, df = 1, p < 0,001$).

5 Conclusões

Neste artigo descrevemos a implementação e avaliação de *PortNLG*, um sistema de realização superficial para o português brasileiro. A partir de uma especificação de entrada abstrata e possivelmente subespecificada, o sistema faz uso de conhecimento linguístico extraído de recursos disponibilizados pela comunidade de PLN do Brasil (em especial, etiquetadores e dicionários) para produzir sentenças em língua natural.

Os resultados aqui discutidos são consideravelmente superiores aos obtidos com uso de modelos estatísticos de n-gramas treinados a partir de uma grande massa de dados. Embora não tenhamos avaliado o tempo de execução do sistema, constatamos que, em ambas modalidades implementadas, a geração de sentenças não consome tempo significativo, o que torna o sistema especialmente atraente para aplicações de tempo real.

Por outro lado, os resultados apresentados são ainda inferiores aos observados em estudos mais recentes com modelos estatísticos ditos fatorados ou FLMs [34]. Cabe destacar entretanto que o uso de FLMs para fins de geração de língua natural podem em certos casos enfrentar dificuldades de ordem prática em função do volume de dados integrantes do modelo e, consequentemente, de seu tempo de avaliação. Em [30], por exemplo, este tempo é da ordem de várias horas de execução em uma configuração de hardware típica, o que pode restringir o uso de sistemas baseados em FLMs à aplicações sem interatividade.

Embora o sistema proposto tenha sido testado apenas com base em manchetes de jornais brasileiros, acreditamos que sua aplicação a outros gêneros literários seja relativamente simples, já que a construção da sentença é realizada passo-a-passo a partir de seus componentes mais básicos (ou sintagmas). Basicamente, a única exigência para isso é que os vocábulos daquele domínio sejam disponibilizados na base lexical do sistema.

Pelo mesmo motivo, acreditamos que o sistema possa ser aplicado à maioria dos casos de geração de sentenças também em português europeu. Construções gramaticais não utilizadas no português brasileiro, como o sintagma verbal em 'estou a ler um artigo', são naturalmente contempladas na modalidade de geração incremental. Estas construções podem também ser facilmente reproduzidas na modalidade de geração a partir de árvore com a simples criação de um template para este tipo de sintagma (que no exemplo representaria o uso da preposição 'a' seguido de um verbo no infinitivo). A comprovação destas ponderações permanece no entanto como oportunidade para trabalhos futuros.

Agradecimentos Este trabalho contou com apoio FAPESP e da Universidade de São Paulo.

Referências

[1] Reiter, E. An Architecture for Data-to-Text Systems. In: European Natural Language Generation workshop (ENLG-2007), pp. 97-104 (2007)

[2] Belz, A. Automatic Generation of Weather Forecast Texts using Comprehensive Probabilistic Generation-Space Models. Natural Language Engineering 14 (4), pp. 431-455 (2008)

[3] Portet, F., Reiter, E., Gatt, A., Hunter, J., Sripada, S., Freer, Y., Sykes, C. Automatic Generation of Textual Summaries from Neonatal Intensive Care Data. In: Artificial Intelligence 173, pp.789-816 (2009).

[4] Oliveira, R.L., Novais, E.M., Araujo, R.P.A., Paraboni, I. A Classification-driven Approach to Document Planning. In: Recent Advances in Natural Language Processing (RANLP-2009), pp.324-329 (2009).

[5] Paraboni. I. Generating references in hierarchical domains: the case of document deixis. Information Technology Research Institute (ITRI), University of Brighton. PhD Thesis (2003).

[6] Paraboni, I., Masthoff, J., van Deemter, K. Overspecified reference in hierarchical Domains: measuring the benefits for readers. 4th International Natural Language Generation Conference (INLG-2006) Sydney, Australia, pp.55-62 (2006).

[7] Lucena, D. J., Pereira, D. B., Paraboni, I. From Semantic Properties to Surface Text: the Generation of Domain Object Descriptions. In: Inteligencia Artificial 14(45) pp.48-58 (2010)

[8] McRoy, S., Channarukul, S., Ali, S. An augmented template-based approach to text realization. Natural Language Engineering 9(4) pp.381-420. Cambridge University Press (2003).

[9] Pereira, D. B., Paraboni, I. Statistical Surface Realisation of Portuguese Referring Expressions. 6th International Conference on Natural Language Processing (GoTAL-2008) Gothenburg, Sweden. Lecture Notes in Artificial Intelligence vol. 5221, pp.383-392. Springer-Verlag Berlin Heidelberg (2008).

[10] Santos, F.M.V., Pereira, D.B., Paraboni, I. Rule- based vs. Probabilistic Surface Realisation of Definite Descriptions. VI Workshop on Information and Human Language Technology (TIL-2008). XIV Brazilian Symposium on Multimedia and the Web, pp. 372-374 (2008).

[11] Gatt, A., Reiter, E. SimpleNLG: A realization engine for practical applications. In: European Natural Language Generation workshop (ENLG-2009) pp. 90-93 (2009)

[12] Langkilde, I. Forest-based statistical sentence generation. In: Proceedings of ANLP-NAACL'00, pp. 170-177 (2000)

[13] Langkilde-Geary, I. An empirical verification of coverage and correctness for a general-purpose sentence generator. In: Proceedings of the International Natural Language Generation Conference (INLG-2002), pp. 17-24 (2002)

[14] Oh, A., Rudnicky, A. Stochastic language generation for spoken dialogue systems. In: Procs. of the ANLP-NAACL'00 Workshop on Conversational Systems, pp. 27-32 (2000)

[15] Novais, E.M., Tadeu, T.D., Paraboni, I. Improved Text Generation using N-gram Statistics. 12^{nd} Ibero-American Conference on Artificial Intelligence (IBERAMIA-2010) LNAI vol. 6433, pp. 316-325 Springer-Verlag Berlin Heidelberg (2010)

[16] Novais, E.M., Paraboni, I., da Silva Junior, D. F. P. Portuguese Text Generation from Large Corpora. 8^{th} International Conference on Language Resources and Evaluation (LREC-2012) Istanbul, pp. 4010-4014 (2012)

[17] Reiter, E., Sripada, S. Human Variation and Lexical Choice. Computational Linguistics 28 (4), pp. 545-553 (2002)

[18] Bangalore, S., Rambow, O. Corpus-based lexical choice in natural language generation. In: 38^{th} Meeting of the ACL, Hong Kong, pp. 464-471 (2000)

[19] Belz, A., White, M., Espinosa, D., Kow, E., Hogan, D., Stent, A. The First Surface Realisation Shared Task: Overview and Evaluation Results. In: Proceedings of the 13^{th} European Workshop on Natural Language Generation, pp. 217-226 (2011)

[20] Belz, A., Bohnet, B., Mille, S., Wanner, L., White, M. The Surface Realisation Task: Recent Developments and Future Plans. In: Proceedings of the 7^{th} International Natural Language Generation Conference (INLG-2012), pp. 136-140 (2012)

[21] Guo, Y., Hogan, D., van Genabith, J. DCU$*$at Generation Challenges 2011 Surface Realisation Track. In: Proceedings of the 13^{th} European Workshop on Natural Language Generation, pp. 227-229 (2011)

[22] Stent, A. ATT-0: Submission to Generation Challenges 2011 Surface Realization Shared Task. In: Proceedings of the 13^{th} European Workshop on Natural Language Generation, pp. 230-231 (2011)

[23] Gervas, P. UCM Submission to the Surface Realization Challenge. In: Proceedings of the 13^{th} European Workshop on Natural Language Generation, pp. 239-241 (2011)

[24] Muniz, M.C.M. A construção de recursos linguístico-computacionais para o português do Brasil: o projeto de Unitex-PB. Msc. dissertation. ICMC / USP, 72 p. (2004)

[25] Aluisio, S. M. , Specia, L., Pardo, T.A.S., Maziero, E., Fortes, R.P.M. Towards Brazilian Portuguese Automatic Text Simplification Systems. The ACM Symposium on Document Engineering, pp.240-248 (2008).

[26] Aziz, W. F., Pardo, T.A.S., Paraboni, I. Statistical Phrase-based Machine Translation: Experiments with Brazilian Portuguese. In: Proceedings of ENIA-2009, pp.769-778 (2009).

[27] Aziz, W. F., Pardo, T.A.S., Paraboni, I. An Experiment in Spanish-Portuguese Statistical Machine Translation, In: Proceedings of SBIA-2008, pp.248-257 (2008).

[28] de Novais, E.M., Paraboni, I. Portuguese text generation using factored language models. Journal of the Brazilian Computer Society 19(2) pp.135-146. Springer-Verlag (2012).

[29] Pereira, D.B., Paraboni, I. A Language Modelling Tool for Statistical NLP. In: Proceedings of TIL-2007, pp.1679-1688 (2007).

[30] Novais, E.M., Paraboni, I. Highly-inflected Language Generation using Factored Language Models. 12^{nd} International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2011) LNCS 6608, pp. 429-438, Springer-Verlag Berlin Heidelberg. (2011)

[31] Nunes, M.G.V., Vieira, F.M.C., Zavaglia, C., Sossolote, C.R.C., Hernandez, J. A construção de um léxico para o português do Brasil: lições aprendidas e perspectivas. II PROPOR, pp. 61-70 (1996)

[32] NIST: Automatic Evaluation of Machine Translation Quality using n-gram Co-occurrence Statistics.[9] (2002)

[33] Papineni, S., Roukos, T., Ward, W., Zhu, W. Bleu: a method for automatic evaluation of machine translation. In: ACL-2002, pp. 311-318 (2002)

[34] Bilmes, J., Kirchhoff, K. Factored Language Models and Generalized Parallel Backoff. In: Proceedings of HLT-NAACL-2003 vol. 2, pp. 4-6 (2003)

[9] www.nist.gov/speech/tests/mt/doc/ngram-study.pdf