UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

APRENDIZADO DE MÁQUINA MULTIVISÃO APLICADO À ANÁLISE DE CORREFERÊNCIA EM UM SISTEMA DE APRENDIZADO SEM FIM

Alex Fernandes Mansano

Orientador: Estevam Rafael Hruschka Junior

São Carlos – SP

Fevereiro/2018

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

APRENDIZADO DE MÁQUINA MULTIVISÃO APLICADO À ANÁLISE DE CORREFERÊNCIA EM UM SISTEMA DE APRENDIZADO SEM FIM

Alex Fernandes Mansano

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Aprendizado de Máquina e Mineração de Dados
Orientador: Estevam Rafael Hruschka Junior

São Carlos – SP

Fevereiro/2018

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de Dissertação de Mestrado do(a) candidato(a) **Alexandre Fernandes Mansano**, realizada em **01 de março de 2018.**

****

Prof(a). Dr(a). Estevam Rafael Hruschka Junior
(UFSCar)

Alexandre Luis M. Levada

Prof(a). Dr(a). Alexandre L. Magalhães Levada
(UFSCar)

****

Prof(a). Dr(a). Jaime dos Santos Cardoso
(FEUP)

Certifico que a sessão de defesa foi realizada com a participação à distância dos membros Jaime dos Santos Cardoso e Estevam Rafael Hruschka Junior, depois das arguições e deliberações realizadas, o participante à distância está de acordo com o conteúdo do parecer da comissão examinadora redigido no relatório de defesa do(a) aluno(a).

Alexandre Luis M. Levada

Prof(a). Dr(a). Alexandre L. Magalhães Levada
(UFSCar)

O que sabemos é uma gota; o que ignoramos é um oceano.

Isaac Newton

Resumo

A NELL (*Nerver-Ending Language Learning*) é o primeiro sistema de aprendizado sem fim presente na literatura. Este sistema foi modelado para criar uma base de conhecimento autônoma, lendo a Web 24 horas por dia, sete dias por semana. Neste paradigma, todo conhecimento adquirido é utilizado para melhorar sua performance. Diante deste paradigma nos deparamos em casos onde um mesmo objeto pode ser nomeado de diversas maneiras. Estes casos são denominados como correferentes e têm grande importância para o aprendizado sem fim, pois o conhecimento sobre certa entidade em uma base textual pode estar distribuído entre suas diversas denominações. Desta forma, a análise de correferência tem um papel crucial no paradigma de aprendizado da NELL.

Neste projeto, nós abordamos técnicas de aprendizado multivisão, combinando diferentes vetores de características como uma tarefa de otimização, executado por técnicas meta-heurísticas e redes neurais artificiais, a fim de maximizar a separabilidade das amostras no espaço amostral, sendo que o processo de otimização é guiado pela acurácia dos classificadores Floresta de Caminhos Ótimos e variantes de Redes Neurais Siamesas em um conjunto de validação. Os experimentos mostraram que a metodologia proposta pode obter resultados melhores quando comparado à performance de extração de características individuais.

Palavras-chave: Never-Ending Language Learning, Meta-heurística, Combinação de Descritores, Redes Neurais

Abstract

NELL (Never-Ending Language Learning) is the first never-ending learning system presented in the litera ture. It has been modeled to create a knowledge base in an autonomous way, reading the web 24 hours per day, seven days per week. In this paradigm, all knowledge acquired is used to improve the learning performance.
In this paradigm we face cases where the same object can be named in several ways. These cases as called as correferents, and has great importance for the never-ending learning process, as long as the knowledge about certain entity in a textual base may be distributed among its denominations.As such, the co-reference analysis has a crucial role in NELLs learning paradigm.
In this paper, we approach the combination of different feature vectors as an optimization task performed by meta-heuristic techniques and artificial neural networks, in order to maximize the separability of samples in the feature space, being the optimization process guided by the accuracy of Optimum Path Forest and variations of Siamese Networks in a validation set. The experiments showed the proposed methodology can obtain much better results when compared to the performance of individual feature extraction algorithms.

Keywords: Never-Ending Language Learning, Meta-heuristics, Descriptor Combination, Neural Networks

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

1.1 Contexto

A NELL[1] (Never-Ending Language Learner) (CARLSON et al., 2010; MITCHELL et al., 2015) é um sistema computacional de aprendizado sem fim, em operação 24 horas por dia, 7 dias por semana desde janeiro de 2010. Tal sistema possui como principal objetivo aprender a melhorar (continuamente) sua capacidade de leitura. Assim, a NELL busca aprender a ler cada vez melhor, para conseguir aprender cada vez mais, através da leitura. Em sua busca diária pelo aprendizado e pela extração de conhecimento (a partir da Web), este sistema de aprendizado sem-fim (a NELL) extrai fatos relacionados a categorias e relações, as quais são previamente fornecidas como entrada do sistema. Estas categorias e relações, inicialmente fornecidas, formam uma ontologia inicial e guiam o sistema em seu aprendizado. Na ontologia inicial da NELL, *pessoa*, *cidade*, *empresa* são possíveis exemplos de categorias. Como exemplos de possíveis relações (entre as categorias) podemos ter, por exemplo, *trabalhaPara(pessoa, empresa)*, *prefeitoDe(pessoa, cidade)*, *sediadaEm(empresa, cidade)*, etc.

Os fatos extraídos a partir da leitura são automaticamente armazenados na Base de Conhecimento (BC) da NELL, num processo chamado de População da Base de Conhecimento. A BC da NELL é composta, basicamente, por *conceitos* e *relações*. Os conceitos da BC estão diretamente ligados às categorias, sendo que um conceito é uma instância de uma categoria. Neste sentido, pode-se ter, por exemplo, o predicado unário *cidade(São Carlos)* como sendo um conceito, que representa uma instância da categoria *cidade*. Seguindo o mesmo raciocínio, uma relação é uma instância (formada por um par de conceitos) de uma relação definida na ontologia inicial. Assim, um possível exemplo

[1] http://rtw.ml.cmu.edu

de relação pode ser representado, por exemplo, pelo predicado binário *sediadaEm(Tapetes São Carlos, São Carlos)* como sendo uma instância de *sediadaEm(empresa, cidade)*.

A tarefa de População da Base de Conhecimento da NELL se iniciou com base apenas em textos escritos na língua inglesa. Assim, considere por exemplo, a seguinte sentença ($s1$):

$s1$: ***De Blasio*** *was elected the mayor of* ***New York***.

Para popular sua BC, quando a NELL lê a sentença $s1$, ela deve ser capaz de mapear o sintagma nominal **"De Blasio"** com o conceito **person(Bill de Blasio)**, o qual é uma instância da categoria *person*. Deve ser capaz também, de mapear o sintagma nominal **"New York"** com o conceito **city(New York City)**, o qual é uma instância da catogoria *city*. O mesmo princípio deve ser seguido para o mapeamento do predicado binário *MayorOf(**Bill De Blasio**,**New York City**)*.

A representação com base em conceitos permite que a NELL possa lidar com situações em que um mesmo conceito pode ser representado por diferentes sintagmas nominais, situação conhecida como **sinonímia** ou **correferência**. Para ilustrar um caso de sinonímia, considere novamente o conceito **city(New York City)** (presente no exemplo da sentença $s1$). Este conceito pode ser representado de diversas maneiras diferentes em várias sentenças diferentes (disponíveis em textos da Web), como por exemplo (incluindo o exemplo da sentença $s1$):

- $s1$: ***De Blasio*** *was elected the mayor of* ***New York***.
- $s2$: ***NYC*** *has architecturally noteworthy buildings.*
- $s3$: ***New York City*** *contains the headquarters of many major corporations.*

Para manter a consistência de sua BC, bem como para poder melhorar sua capacidade de leitura, é importante que um sistema de aprendizado sem fim, como a NELL, consiga identificar situações de sinonímia e realizar o correto mapeamento de diferentes sintagmas nominais para um único conceito. Neste sentido, considerando-se as sentenças $s1$, $s2$ e $s3$, dadas como exemplo, o conceito *city(New York City)* foi representado utilizando-se três formas distintas: ***"New York"***, ***"NYC"*** e ***"New York City"***, sendo que todas elas devem ser mapeadas para o mesmo conceito. Há também os casos de polissemia, em que um sintagma nominal é capaz de designar diversos conceitos, como *Apple* pode representar a fruta "maçã"em inglês ou mesmo a empresa de tecnologia "Apple Inc".

O trabalho descrito em (KRISHNAMURTHY; MITCHELL, 2011) apresenta com mais detalhes a representação da base de conhecimento, baseada em conceitos, utilizada pela NELL, para lidar com a sinonímia, bem como com a polissemia[2]. Ainda no mesmo trabalho (KRISHNAMURTHY; MITCHELL, 2011), é proposto uma abordagem, chamada *ConceptResolver*, projetada para identificar (e mapear de maneira correta) tanto as situações de sinonímia, quanto as de polissemia. Atualmente, o ConceptResolver é utilizado pela NELL como abordagem padrão na identificação e correção de situações de correferência.

No trabalho descrito em (DUARTE; Hruschka Jr., 2014) é realizada uma análise sobre o ConcetpResolver, e uma das principais limitações identificadas está relacionada ao fato de a base de dados da NELL não ser completa. Ou seja, por ser um sistema de aprendizado contínuo e sem fim, é natural que novos conhecimentos estejam sempre surgindo. Sendo natural também, que muitos fatos não tenham ainda sido aprendidos (e armazenados na BC). Por este motivo, é muito comum a presença de valores ausentes na BC. Um exemplo de valor ausente pode ser, por exemplo, a NELL ainda não ter aprendido quem é o prefeito de uma cidade. Desta forma, ao encontrar os valores ausentes o método encontra dificuldades em identificar correferência.

Buscando atenuar as limitações do ConceptResolver, foi proposta (veja (DUARTE; Hruschka Jr., 2014)) uma abordagem de aprendizado de máquina de multivisão que busca explorar atributos morfossintáticos (visão 1), e também atributos presentes na representação interna da BC da NELL (visão 2). Tal proposta foi definida como uma etapa posterior ao ConceptResolver e não como uma substituição do método de tratamento de correferência. Neste sentido, a saída do ConceptResolver é usada como entrada da abordagem multivisão e o processo de aprendizado de máquina pode auxiliar a melhorar o desempenho na identificação e tratamento de correferências. Mesmo tendo atingido resultados importantes, a abordagem multivisão proposta em (DUARTE; Hruschka Jr., 2014) não consegue trazer ganho significativo em casos envolvendo áreas da BC contendo muitos valores ausentes. Mais detalhes sobre o trabalho multivisão são dados na Seção 2.12.

1.2 Motivação e Objetivos

Com base nas características específicas de um sistema de aprendizado sem-fim, conforme descrito na seção anterior, o problema de identificação e tratamento de correferência é relevante. Além disso, a abordagem a ser utilizada no tratamento de correferência da

[2]neste trabalho não será abordada a busca por soluções para a polissemia.

NELL não deve ter como base apenas as abordagens padrão, definidas no domínio do *Processamento de Língua Natural* (PLN).

Na proposta multivisão definida em (DUARTE; Hruschka Jr., 2014), por exemplo, a visão 1 é definida com um conjunto de atributos mais comumente utilizados em abordagens no domínio do PLN, ou seja, utiliza-se atributos morfo-sintáticos para se treinar um classificador binário capaz de identificar correferências em que dois sintagmas nominais sejam compostos por cadeias de caracteres similares. Como exemplo, a visão 1 teria maior facilidade para identificar correferências como: *New York City* e *NYC*. Entretanto, este tipo de atributos tende a não ser muito adequado na identificação de pares de sintagmas nominais correferentes, como por exemplo: *New York City* e *Big Apple*.

Já a visão 2, na mesma proposta de Duarte (2014), tem como base o vetor de características automaticamente criado pela NELL para representar cada conceito aprendido a partir da leitura sem-fim. Neste sentido, o conceito *city(New York City)* e o conceito *city(Big Apple)* possuem similaridade em relação, por exemplo, às relações *prefeitoDe(New York City, De Blasio)* e *prefeitoDe(Big Apple, De Blasio)*, bem como em *cidadeLocalizadaNoEstado(New York City, New York)* e *cidadeLocalizadaNoEstado(Big Apple, New York)*.

Uma terceira visão empregada é a utilização de informações provenientes de outras fontes, além da NELL. Neste contexto também é proposta a utilização de técnicas de extração de características de grandes bases de texto, como o *corpus*, ou córpora, da *Wikipedia*. Desta forma, também são utilizadas fontes de dados externas à NELL para criar representações vetoriais de palavras e capturar informações semânticas das mesmas, de forma que as palavras *Big Apple* e *New York* sejam próximas em um determinado espaço vetorial. Uma vantagem desta técnica reside no fato de, dado um conjunto de dados suficientemente grande, não ser necessário retreinar o modelo, uma vez que gerado todas as palavras já constarão no vocabulário.

É possível notar que, intuitivamente, as duas visões (acima mencionadas) tendem a ser complementares, e por isso, tendem a ser boas opções na proposta de uma abordagem classificação multivisão. A característica de aprendizado sem-fim, intrínseca à NELL, traz entretanto, a possibilidade de muitos valores ausentes. Com isso, a visão 2 pode ter seu desempenho prejudicado. Com base neste problema, esta proposta de trabalho de mestrado apresenta a seguinte hipótese de pesquisa:

Hipótese: A utilização de uma terceira visão, a qual tenha como base um conjunto de atributos extraídos de um conjunto de dados textual (*corpus*)

suficientemente grande, pode contribuir na melhoria da qualidade da identificação e do tratamento de correferência na NELL, quando utilizada juntamente com as duas visões já propostas em abordagens anteriores.

O principal objetivo do trabalho de mestrado, é a investigação e a análise da hipótese de pesquisa (acima definida), buscando-se formas que permitam a obtenção de evidências empíricas capazes de apontar indícios para uma tendência de sua aceitação ou refutação. Desta forma, o presente trabalho apresenta:

1) Investigação e aplicação de abordagens adequadas para a obtenção de um conjunto de atributos apropriados à terceira visão a ser definida para o problema de identificação e tratamento de correferência; 2) Investigação e aplicação classificadores multivisão ao problema, explorando-se a combinação de diferentes visões.

1.3 Estrutura e Organização deste Documento

A sequência deste documento está organizada da seguinte forma: No Capítulo 2 é explicado o conceito e estrutura de um sistema de aprendizado sem fim, o modelo de representação distribuída de palavras e abordagens para análise de correfêrenica. Neste capítulo também são apresentados os algoritmos evolutivos utilizados no processo de otimização para análise de correferência bem como o Classificador Floresta de Caminhos Ótimos e técnicas de aprendizado neural, empregados para a tarefa de classificação dos pares de frases nominais em correferentes ou não correferentes. A seção 2.12 apresenta trabalhos relacionados que embasaram o presente projeto.

O Capítulo 3 descreve os objetivos e a visão empregada neste trabalho. Os experimentos, resultados e análise são discutidos nas seções 3.2 e 3.3 deste capítulo.

Por fim, o Capítulo 4 apresenta a conclusão e trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Never Ending Language Learner

Mitchel et al. (MITCHELL et al., 2015) definem um agente de aprendizado sem fim[1] como um sistema que é capaz de, assim como os humanos, aprender diversos tipos de conhecimento a partir de anos de de experiências auto-supervisionadas, utilizando conhecimentos obtidos no passado para melhor seu aprendizado subsequente.

Um problema de aprendizado sem fim ρ é representado por uma tupla, que consiste em um conjunto $L = \{L_i\}$ de tarefas de aprendizado, sendo que a tarefa $L_i = \langle T_i, P_i, E_i \rangle$ melhora a performance dos agentes medida através da métrica P_i em uma tarefa T_i de acordo com uma certa experiência E_i e um conjunto de restrições $Q = \{\langle \phi_k, V_k \rangle\}$ dentre as soluções para a tarefa dada, onde ϕ_k é uma função real sobre duas ou mais tarefas que especifica o grau de satisfação da restrição proposta e V_k é o vetor de índices das tarefas a que estão sendo aprendidas, especificando-se os argumentos ϕ_k, sendo que F_i é o conjunto de todas as possíveis funções de X_i e Y_i:

$$\begin{aligned} \rho &= (L, Q) \\ \text{onde, } L &= \{\langle T_i, P_i, E_i \rangle\} \ . \\ Q &= \{\langle \phi_k, V_k \rangle\} \end{aligned} \tag{2.1}$$

Cada tarefa de performance $T_i \equiv \langle X_i, Y_i \rangle$ define o domínio e os limites da função a ser aprendida $f_i^* : X_i \to Y_i$. A métrica de performance $P_i : f \to \Re$ define a função ótima aprendida $f_i^* \equiv \underset{f \in F_i}{\operatorname{argmax}} P_i(f)$ para a i-ésima tarefa de aprendizagem.

Dado o problema contendo n tarefas de aprendizagem, um agente de aprendizado

[1] Never Ending Learner

sem fim ω gera um conjunto de soluções para estas tarefas. Conforme o tempo passa, a qualidade dessas funções deve melhorar quando medida de acordo com as métricas $P_1, \ldots, P_n$ e as restrições Q são respeitadas.

O sistema de aprendizado sem fim aqui apresentado é o *Never Ending Language Learner*, conhecido como NELL, que pode ser visto como um sistema de computador que roda 24 horas por dia, 7 dias na semana, para sempre a fim de popular sua base de conhecimento. De acordo com Carlson et al. (CARLSON et al., 2010) duas tarefas são realizadas diariamente:

- **Tarefa de leitura**: extrai informações de textos da Web para popular sua crescente base de fatos estruturados e conhecimento.
- **Tarefa de aprendizagem**: aprende a ler cada dia melhor que o dia anterior, graças à sua capacidade de poder voltar às fontes de dados do dia anterior e extrair mais informações de forma mais precisa.

A NELL adquire dois tipos de conhecimento:

1. conhecimento sobre quais frases nominais[2], ou sintágmas, referem-se a quais categorias semânticas[3];
2. conhecimento sobre quais pares de frases nominais especificam quais relações[4] semânticas.

A NELL aprende a obter esses dados de diversas maneiras, como lendo padrões de textos não formatados, sentenças da Internet, dados estruturados da internet, como tabelas e listas, além de aprender regularidades morfológicas de instâncias de categorias e cláusulas de Horn[5] que a possibilita inferir novas instâncias de relações a partir das relações que ela já conhece.

As categorias e relações adicionadas à base de dados da NELL são divididas em *candidatos a fato* e *crenças*. Os subsistemas leem a base de dados e consultam fontes de conhecimento externas, como corpus textuais e a Internet para em seguida propor novos candidatos a fato. O integrador de conhecimento examina os candidatos a fatos propostos

[2]Frase que não apresenta verbos.
[3]ex: cidades, empresas, esporte, times esportivos...
[4]Relações do tipo: TemSedeEm(Empresa,Cidade)
[5]Cláusula com no máximo um literal positivo, ex: $\neg p, \neg q, \cdots, \neg t \rightarrow r$.

e as evidências que os apoiam e selecionam os com evidências fortes para se tornarem crenças.

A base de dados da NELL pode ser brevemente descrita como uma base de conhecimento ontológico (*Ontological knowledge base - OKB*), a qual pode ser representada por um grafo direcionado rotulado, onde os nós são compostos por entidades x e cada aresta de nome R é uma relação ou regra do tipo $R(x_1,x_2)$ (APPEL; Hruschka Jr, 2011). Uma OKB armazena informações através de um modelo ontológico composto por categorias, relações e fatos[6]. A Figura 2.18 ilustra um exemplo de OKB.

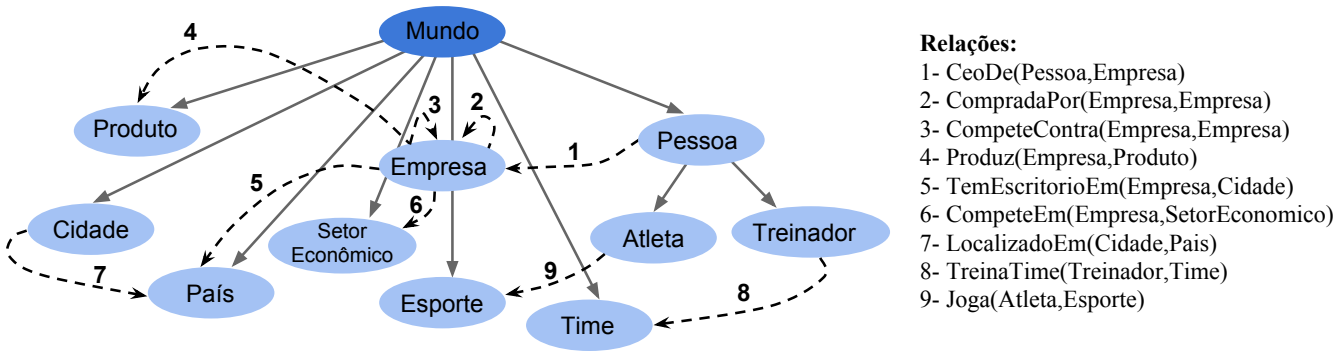


Figura 2.1: Exemplo de um subconjunto da base de conhecimentos da NELL. Adaptado de (APPEL; Hruschka Jr, 2011).

A NELL também utiliza quatro componentes como subsistemas:

- *Coupled Pattern Learner (CPL)*: extrator baseado em textos que aprende e utiliza padrões contextuais do tipo "prefeito de X", "X joga para Y" para extrair instâncias de categorias e relações.
- *Coupled SEAL (CSEAL)*: extrator semi-estruturado que vasculha a Internet com conjuntos de crenças de cada categoria ou relação, e então minera listas e tabelas para extrair novas instâncias do predicado[7] correspondente.
- *Coupled Morphological Classifier (CMC)*: um conjunto de modelos de regressões logísticas binárias, que classifica frases nominais baseando-se em características morfológicas.
- *Rule Learner (RL)* algoritmo de aprendizado de primeira ordem similar ao FOIL (QUINLAN, 1990) que aprende cláusulas de Horn probabilísticas, as quais são utilizadas para inferir novas instâncias.

[6]Instâncias de categorias, ex: **empresa**(*Google*).

[7]Função booleana. Uma declaração que deve ser verdadeira ou falsa dependendo do valor de suas variáveis.

A arquitetura simplificada da NELL pode ser observada abaixo:

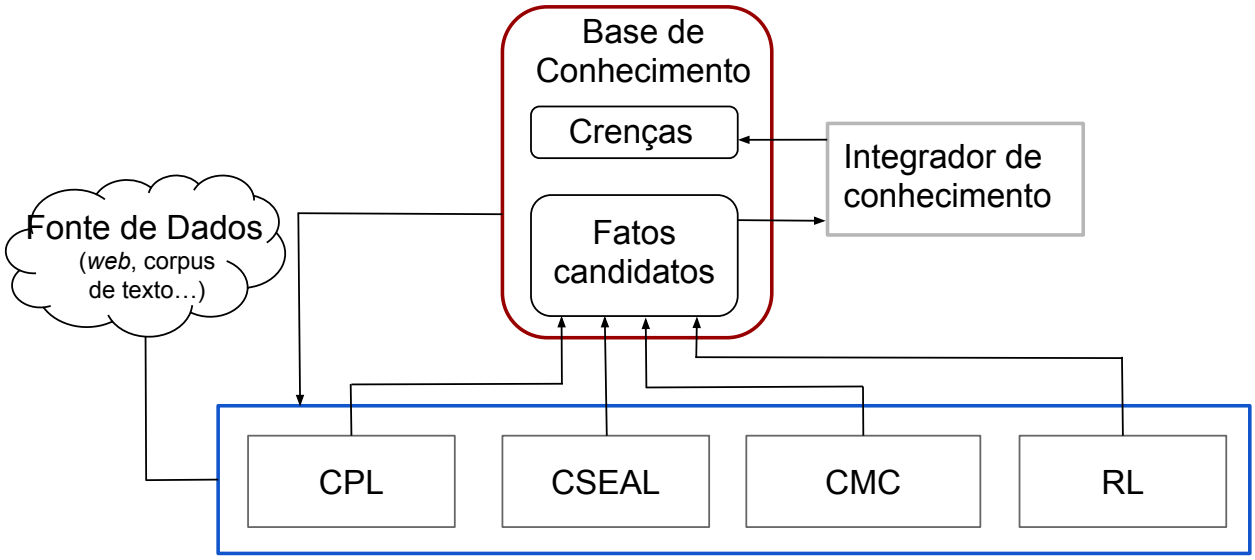


Figura 2.2: Arquitetura original da NELL proposta em (CARLSON et al., 2010).

2.2 *Metric Learning*

A performance de muitos algoritmos de aprendizado de máquina e mineração de dados depende criticamente de uma representação bem definida dos dados em um espaço métrico, que reflita as relações e semelhança entre os dados (XING et al., 2003). Em muitos problemas, especificar matematicamente a relação de semelhança entre os exemplos é muito difícil, por ser muito complexo desenvolver uma função métrica de distância entre eles em um espaço multidimensional. Na classificação de documentos, por exemplo, textos podem ser classificados como similares de acordo com o estilo de escrita e não apenas de acordo com o conteúdo. Para este caso, o algoritmo deveria aprender quais são as características críticas para tais semelhanças e como elas se relacionam.

A fim de resolver este problema, técnicas de aprendizado de máquina buscam aprender essas informações a partir de um conjunto de dados (grande parte das vezes, pares de pontos no espaço $\mathbb{R}^n$) rotulados como similares ou não similares, gerando uma métrica de distância do tipo $d : \mathbb{R}^n \times \mathbb{R}^n :\rightarrow \mathbb{R}$ capaz de mapear para um número real a similaridade dos exemplos no espaço inicial.

De acordo com (BELLET; HABRARD; SEBBAN, 2013), o principal objetivo de uma técnica de *Metric Learning* é adaptar uma função métrica de pares, e.g. distância de

Mahalanobis entre dois pontos x_1 e x_2 sendo $d_M(x_1,x_2) = \sqrt{(x_1-x_2)^T M(x_1-x_2)}$ para o problema a ser tratado utilizando os exemplos de treinamento.

A Maioria dos métodos aprende a métrica necessária (neste caso a matriz positiva semi-definida M) de modo supervisionado a partir de pares de restrições de duas formas:

1. Pares positivos e negativos:

$$S = \{(x_i,x_j) : x_i \text{ e } x_j \text{ são similares}\}, \tag{2.2}$$

$$D = \{(x_i,x_j) : x_i \text{ e } x_j \text{ não são similares}\} \tag{2.3}$$

2. Triplets:

$$R = \{(x_i,x_j,x_k) : x_i \text{ deve ser mais similar a } x_j \text{ do que a } x_k\}, \tag{2.4}$$

Um algoritmo de *Metric Learning* busca encontrar parâmetros para a métrica que melhor se ajusta às restrições acima, aproximando exemplos similares no espaço e distanciando os exemplos distintos. Com isso, mantemos todos os pontos pertencentes à mesma classe próximos e separamos os pontos pertencentes a classes distintas.

2.3 Modelo de Aprendizado Neural

As Redes Neurais Artificiais são modelos computacionais matemáticos baseados no sistema sistema nervoso biológico. São capazes de adquirir conhecimento a partir de experiências passadas e, assim como o cérebro biológico, são compostas por um conjunto de unidades de processamento, os neurônios, que podem ser organizados por um grande número de interconexões.

As redes neurais foram primeiramente empregadas em 1943, por (MCCULLOCH; PITTS, 1943), o qual apresentou a primeira ideia de neurônio artificial. O primeiro método de treinamento para esta técnica foi apresentado em 1949, por Donald O. Hebb (HEBB, 1949), porém o grande interesse na área surgiu apenas em 1958, quando Frank Rosenblatt apresentou o modelo de *Perceptron* (ROSENBLATT, 1958). A Figura 2.3 ilustra um modelo de neurônio.

Cabe notar que o modelo Perceptron só converge quando as classes são linearmente separáveis. Para problemas não linearmente separáveis é necessário utilizar neurônios do tipo perceptron conectados entre si, formando uma rede neural.

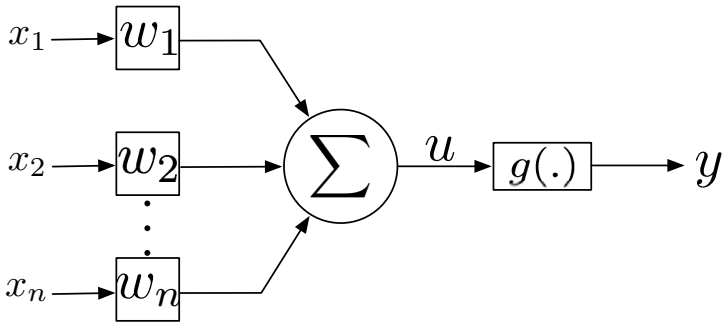


Figura 2.3: Neurônio Artificial. Adaptado de (GOMIDE, 2012).

Os neurônios artificiais presentes nos modelos de redes neurais são responsáveis pela realização de tarefas simples, como receber sinais em suas entradas x_i, agregá-los por meio de um combinador linear $\sum$ e produzir uma saída y, de acordo com a função de ativação $g(.)$ empregada, que objetiva limitar a saída do neurônio dentro de um intervalo numérico. Os valores $w_1, w_2, \ldots, w_n$ são os pesos sinápticos, os quais ponderam cada variável de entrada x_i da rede, regulando sua relevância ao respectivo neurônio.

A forma com que os neurônios são arranjados dentro de uma rede neural é definida pela sua arquitetura. De acordo com (GOMIDE, 2012), uma rede neural pode ser dividida em camadas, as quais são definidas como:

- **Camada de Entrada**: responsável pelo recebimento de informações e sinais externos;
- **Camadas Escondidas ou Intermediárias**: são compostas por neurônios responsáveis pela extração de características referente ao processo de inferência ou aprendizagem, onde ocorre a maior parte do processamento da rede;
- **Camada de Saída**: responsável pela produção e apresentação dos resultados finais da rede, a partir dos dados processados pelos neurônios das camadas escondidas.

Para o desenvolvimento do presente projeto de pesquisa, duas arquiteturas de redes neurais são essenciais, o modelo *feedfoward* e as redes recorrentes.

2.3.1 Modelo *feedfoward*

O modelo de rede *feedfoward* é formado por neurônios interconectados, de forma que não haja ciclos causados por essa conexão. Assim, a propagação dos sinais de entrada

é realizada em um único sentido, a partir da camada de entrada em direção à camada de saída. Esta arquitetura pode ser apresentada em camada única (*single-layer*) e em multi-camadas (*multi-layer*), como descrito abaixo:

- **Camada única**: este modelo é composto por uma única camada de neurônios, que constitui a própria saída da rede. Uma rede neural de camada única, formada por quatro neurônios, é ilustrada na Figura 2.4;
- **Multi-camadas ou profunda (*deep*)**: uma rede neural multi-camadas é formada por uma ou mais camadas intermediárias, as quais também são denominadas escondidas, pois elas não são vistas pela entrada ou saída da rede (HAYKIN, 2009). Essas camadas tem como entrada a saída dos neurônios das camadas precedentes. A adição de ao menos uma camada intermediária na arquitetura da rede neural torna a rede capaz de mapear qualquer função real contínua (CYBENKO, 1989). Um modelo com uma camada oculta pode ser observado na Figura 2.5.

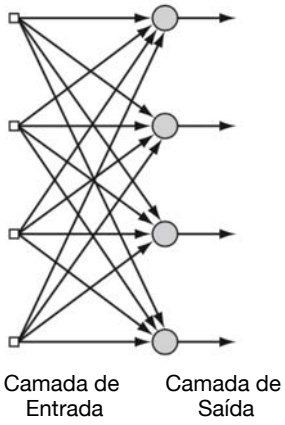


Figura 2.4: Rede neural de camada única. Adaptado de (HAYKIN, 2009).

As redes neurais apresentadas acima são exemplos de redes totalmente conectadas, pois todo nó de cada camada da rede é conectado a todo nó de suas camadas adjacentes.

Ainda dentro do modelo da arquitetura *feedfoward*, este trabalho se foca na aplicação de Redes Neurais Siamesas e Redes Neurais Convolucionais, inicialmente propostas em (BROMLEY et al., 1994) e (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) respectivamente e detalhadas na seção 2.4.

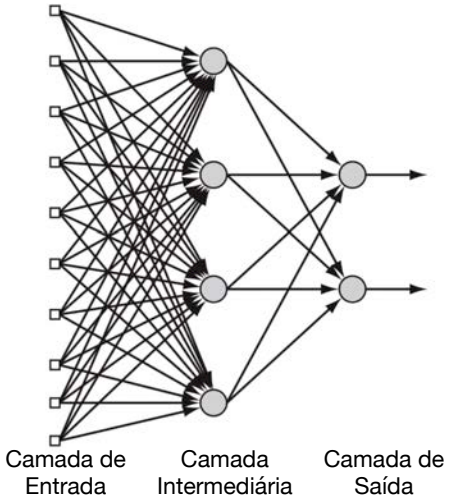


Figura 2.5: Rede neural multi-camadas. Adaptado de (HAYKIN, 2009).

2.3.2 Modelo de Redes Recorrentes

No modelo de rede recorrente, as saídas dos neurônios são realimentadas como sinais de entrada para outros neurônios (GOMIDE, 2012). Essa realimentação possibilita o processamento dinâmico de informações, permitindo o trabalho com variantes temporais, pois as camadas intermediárias podem acessar suas próprias saídas anteriores, moldando seus eventos subsequentes levando em consideração as respostas anteriores, que dá capacidade de memória à rede (ELMAN, 1990). A arquitetura de uma rede recorrente é ilustrada na Figura 2.6.

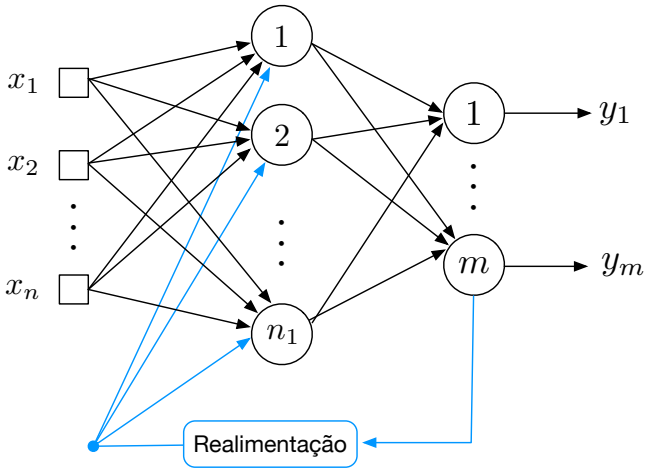


Figura 2.6: Rede neural recorrente. Adaptado de (GOMIDE, 2012).

2.4 Aprendizado profundo - *deep learning*

Nos últimos anos, o modelo Perceptron multi-camadas tem sido mais comumente denominado como rede neural profunda, devido ao aprofundamento do modelo dado pela adição de novas camadas intermediárias à rede.

O objetivo deste modelo consiste, como na maioria dos modelos de aprendizado de máquina, em aproximar uma função f^*. Por exemplo, um classificador $y = f(x)$ que busca mapear uma entrada x para uma categoria y é dado como um mapeamento $y = f(x;\theta)$, no qual a rede aprende os valores de θ, que são os pesos sinápticos da mesma. Neste caso, uma rede com duas camadas intermediárias pode ser vista como uma função $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$, onde cada função $f^{(i)}(.)$ corresponde a uma camada da rede (GOODFELLOW; BENGIO; COURVILLE, 2016), conforme ilustrado abaixo.

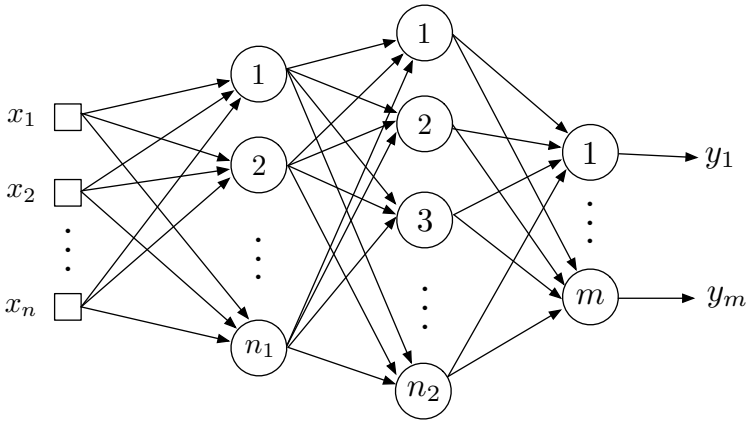


Figura 2.7: Rede neural profunda.

O processo de ajuste de pesos dos neurônios é realizado por meio de treinamento supervisionado, onde para cada amostra dos dados obtém-se uma uma respectiva saída desejada, com auxílio do algoritmo de retropropagação do erro, ou *backpropagation*.

O *backpropagation* pode ser dividido em duas fases, a propagação adiante (*forward*), onde os sinais de entrada são processados e propagados até a camada de saída, e a fase de propagação reversa (*backward*), onde ocorrem os ajustes nos pesos sinápticos em relação ao erro (diferença entre a saída obtida e desejada) da rede.

Dentro do escopo das redes neurais profundas, nos concentramos em duas arquiteturas, as Redes Neurais Siamesas e as Redes Neurais Convolucionais, descritas a seguir.

2.4.1 Redes Neurais Siamesas

As Redes Neurais Siamesas possuem dois campos de entrada, ou seja, recebem dois vetores de características distintos para comparar dois padrões e tem como saída a classificação do par recebido com base na semelhança entre eles em um novo espaço dimensional. As Redes Neurais Siamesas também podem ser vistas como um algoritmo de *metric learning*, discutido na Sessão 2.2.

A arquitetura siamesa é composta por duas redes neurais de mesma topologia e pesos compartilhados, i.e. os pesos empregados em uma rede tem o mesmo valor dos pesos empregados na outra. Estas são unidas por uma função de semelhança $s(x;y)$, onde x e y são os vetores de entrada transformados em um novo espaço vetorial pelas redes que compõem a arquitetura e s é a função de semelhança que computa o quão distantes são as duas instâncias no espaço gerado, o que garante que entradas semelhantes não possam ser mapeadas para espaços distintos pelas suas respectivas redes neurais. Uma outra característica deste grupo de redes é que elas são simétricas, ou seja, não importa a ordem a qual as entradas são submetidas à rede, ela sempre retornará o mesmo resultado (KOCH; ZEMEL; SALAKHUTDINOV, 2015). Um exemplo de rede siamesa é ilustrado na Figura 2.8:

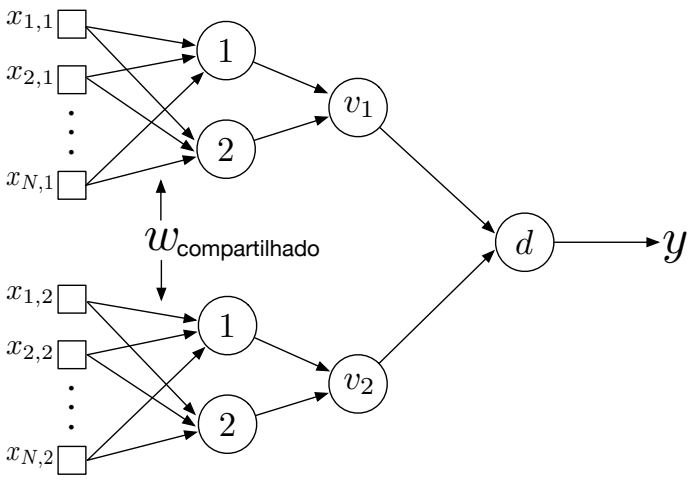


Figura 2.8: Rede Neural Siamesa.

Neste exemplo é apresentada uma rede siamesa com uma camada oculta em cada subrede, as quais mapeiam cada uma das duas entradas nos vetores v_1 e v_2, respectivamente. A distância entre esses vetores é então calculada por meio de uma função de distância d e o par é classificado em uma das possíveis classes pertinentes ao problema. No presente projeto de pesquisa, entre co-referente e não co-referente.

2.4.2 Redes Neurais Convolucionais

Diferentemente das redes apresentadas até até agora, as redes convolucionais não são totalmente conectadas, ou seja, os neurônios de uma camada c_i não estão conectados com todos os neurônios de sua camada precedente $c_(i-1)$. Os modelos convolucionais foram desenvolvidos tomando por base o cortex visual, o qual é composto por milhões de agrupamentos celulares complexos, sensíveis a pequenas sub-regiões do campo visual, chamadas de campos receptíveis (HUBEL; WIESEL, 1968), mais especificamente afim de serem empregadas em problemas de visão computacional, sendo capazes de realizar o reconhecimento de formas bi-dimensionais e invariantes a distorções como translações e escala.

De acordo com Haykin (HAYKIN, 2009), a estrutura de uma rede convolucional pode ser dividida em três objetivos principais:

- **Extração de características**: onde cada neurônio recebe o sinal de entrada de um campo receptível da camada anterior, o que possibilita a extração de características locais. Quando as características são extraídas, sua posição exata deixa de ser importante, desde que sua posição em relação às outras características seja mantida;
- **Mapeamento de características**: cada camada computacional da rede é composta por diversos mapas de características (*feature maps*), que são regiões em um plano onde os neurônios compartilham os mesmos pesos sinápticos, esses pesos compartilhados são conhecidos como filtros, ou *kernels* e permitem ao processo ser robusto em relação a variações de distorção e rotações na imagem, além de reduzir o número de parâmetros livres a serem otimizados.
- **Subamostragem**: após cada camada de convolução, é aplicada uma camada de subamostragem (*subsampling*), que realiza operações de coleta de amostras em cada mapa de característica. Essas amostragens podem ser realizadas obtendo a média da região, selecionando a característica com maior (*max pooling*) ou menor (*min pooling*) valor, somando todos os valores do mapa ou qualquer outra função capaz de sumarizar o mapa de características em questão.

A Figura 2.9 ilustra a arquitetura descrita acima, composta de uma camada de entrada com 28 x 28 sensores e quatro mapas de características de tamanho 24 x 24. Após esta camada, é realizada uma subamostragem que gera mapas de tamanho 12 x 12, às quais

são aplicadas uma nova convolução, com 12 filtros 8 x 8, e uma nova subamostragem. Por fim, são retornados 26 neurônios correspondentes à saída da rede.

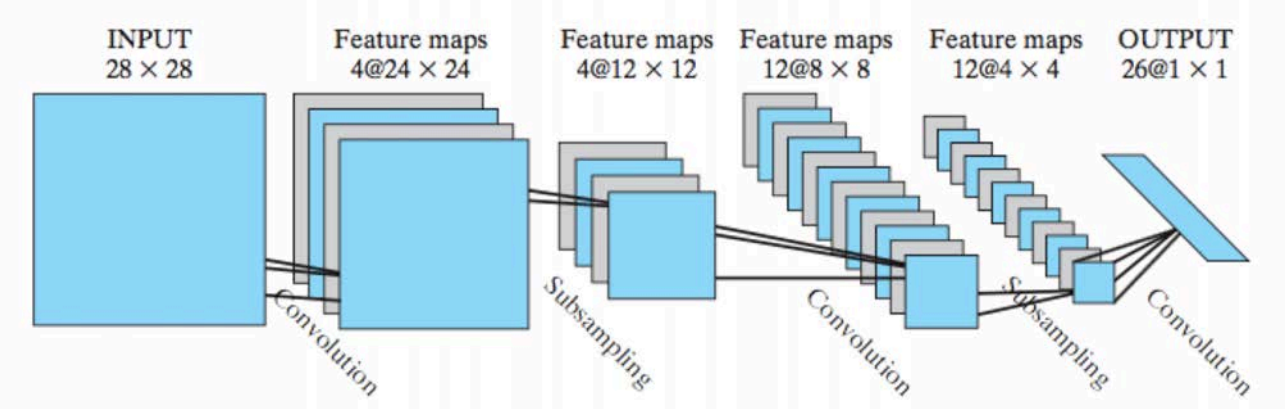


Figura 2.9: Rede Neural Convolucional. Imagem extraída de (HAYKIN, 2009).

A ilustração abaixo exemplifica a aplicação de um filtro 3 x 3 em uma entrada de dimensões 8 x 8, sucedido por uma operação de *max pooling*.

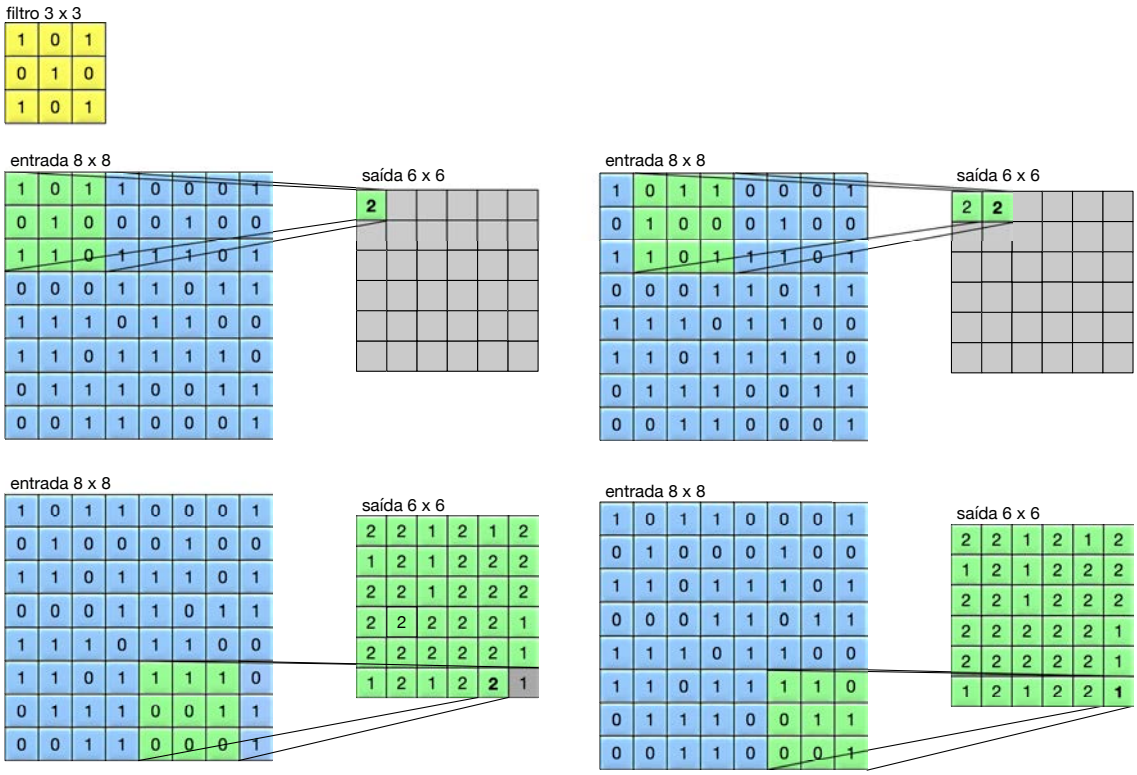


Figura 2.10: Exemplo de filtro aplicado a uma entrada 8 x 8. Nesta operação, cada mapa de característica também de tamanho 8 x 8 é multiplicado pela matriz correspondente ao filtro aplicado. Após isso, é realizada uma operação de *max pooling*, responsável por realizar a subamostragem, selecionando a característica de maior valor dentro do novo mapeamento de características.

Desta forma, o elemento $1,1$ da camada de saída é obtido por:

$$max\left(\begin{bmatrix}1&0&1\\0&1&0\\0&1&0\end{bmatrix}\times\begin{bmatrix}1&0&1\\0&1&0\\1&0&1\end{bmatrix}\right)=2$$

o elemento $1,2$ por

$$max\left(\begin{bmatrix}0&1&1\\1&0&0\\1&0&1\end{bmatrix}\times\begin{bmatrix}1&0&1\\0&1&0\\1&0&1\end{bmatrix}\right)=2$$

sucessivamente, deslizando-se o filtro coluna a coluna e linha a linha até o elemento $8,8$, que é dado pelo cálculo:

$$max\left(\begin{bmatrix}1&1&0\\0&1&1\\0&0&1\end{bmatrix}\times\begin{bmatrix}1&0&1\\0&1&0\\1&0&1\end{bmatrix}\right)=1$$

Os filtros aplicados não possuem necessariamente valores binários. Aqui utilizamos filtros binário para facilitar a compreensão dos cálculos.

2.5 Representação Distribuída de Palavras

Os modelos de representação distribuída de palavras, bem como os vetores de características gerados pelos métodos empregados por (DUARTE; Hruschka Jr., 2014) podem ser vistos como descritores. Torres (TORRES et al., 2009) define um descritor como:

Definição 1 *Um **descritor simples** (brevemente, **descritor**) **D** é definido como um par* $(\varepsilon_D, \delta_D)$*, onde:*

- $\varepsilon_D : np \rightarrow \mathbb{R}^n$ *é uma função que extrai um* vetor de características $\vec{v}_{np}$ *de uma* frase nominal np*.*
- $\delta_D : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ *é uma* função de similaridade *que calcula a similaridade entre duas frases nominais como uma função de distância entre seus correspondentes* vetores de características.

Definição 2 *Um **vetor de característica** $\vec{v}$ é um ponto no espaço* $\mathbb{R}^n$*,* $\vec{v} = (v_1, v_2, ..., v_n)$*, onde* n *é a dimensão do vetor que representa tal objeto.*

Diferentes descritores podem ser combinados, gerando um descritor composto.

Definição 3 *Um* ***descritor composto*** $\hat{D}$ *é um par* $(\mathscr{D}, \delta_{\mathscr{D}})$*, onde:*

- $\mathscr{D} = \{D_1, D_2, \ldots, D_k\}$ *é um conjunto de* k *descritores simples pré-definidos.*
- $\delta^*_{\mathscr{D}}$ *é a função de similaridade que combina os valores de similaridade obtidos por cada descritor* $D_i \in \mathscr{D}$*,* $i = 1, 2, \ldots, k$*.*

A Figura 2.11 ilustra o uso de um descritor simples para o cálculo de similaridade entre duas frases nominais *fn1* e *fn2*. Inicialmente, um algoritmo de extração de características ε_D é utilizado para gerar o vetor de características $\vec{v}_1$ e $\vec{v}_2$ associado a cada frase nominal *fn1* e *fn2*, respectivamente. Então, a função de similaridade δ_D é empregada para encontrar o valor de similaridade d entre *fn1* e *fn2*. A Figura 2.12 ilustra o uso de um descritor composto $\hat{D}$ para computar a distância entre duas frases nominais *fn1* e *fn2*.

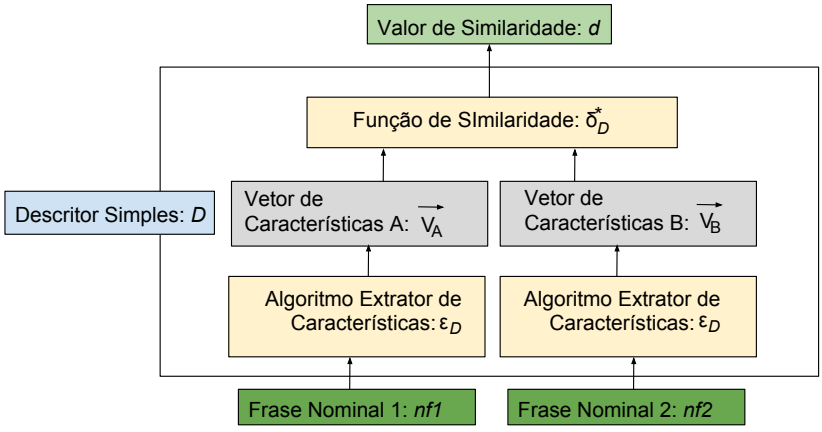


Figura 2.11: Esquema de um descritor simples.

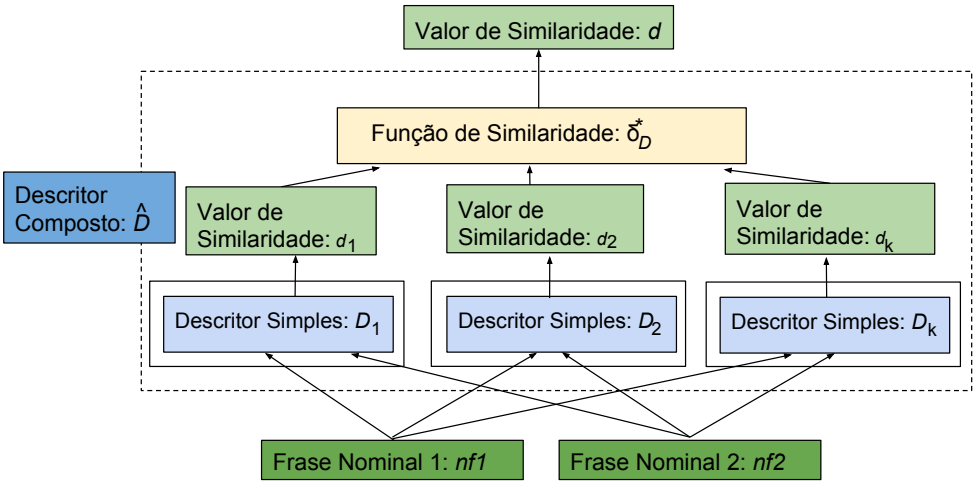


Figura 2.12: Esquema de um descritor composto.

A ideia de representar uma palavra com um vetor de características denso tem se mostrando muito útil nas tarefas de aprendizado de máquina e de processamento de língua natural, como utilizado por (BENGIO et al., 2003; MIKOLOV et al., 2010). A representação distribuída de palavras em um modelo espaço-vetorial auxilia os algoritmos de aprendizado de máquina a alcançar melhor performance em tarefas de processamento de língua natural, por meio de agrupamentos de palavras similares. Ao representarmos uma palavra como um vetor de características, obtemos a possibilidade de calcular a similaridade entre duas frases nominais por meio da distância (e.g. euclidiana ou similaridade de cosseno) entre elas.

Diversos modelos para a representação distribuída de palavras foram propostos no últimos anos, dentre eles destacam-se o proposto por Bengio em "*A Neural Probabilistic Language Model*" (BENGIO et al., 2003); por Mikolov, denominados *Skip-gram*, *CBOW* e *Subword CBOW* (MIKOLOV et al., 2013a; BOJANOWSKI et al., 2016); e o GloVe, proposto por Pennington (PENNINGTON; SOCHER; MANNING, 2014). Estes métodos propõem a construção de um modelo neural para a criação de uma representação distribuída de palavras, cuja ideia central parte da combinação de vetores no espaço semântico com modelos de previsões probabilísticas. Nestes modelos as palavras são representadas como um vetor de características, e palavras similares são representadas por vetores similares, conforme apresentado na Figura 2.13.

O objetivo desses modelos é computar por meio de uma rede neural quais os valores do vetor de características de cada palavra. Alguns deles capturam melhor representações sintáticas, como o *Subword CBOW* e outros informações semânticas, que é o caso do

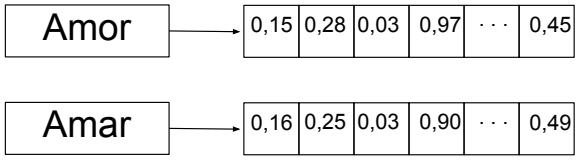


Figura 2.13: Representação distribuída da palavra Amor.

GloVe. Estes modelos são descritos nas sub-seções seguintes.

2.5.1 *Modelo de Linguagem Neuro Probabilístico*

Em seu modelo, Bengio (BENGIO et al., 2003) propõe criar uma representação vetorial de palavras por meio de uma rede neural de três camadas, onde os dados de treinamento são formados por uma sequência $w_1, w_2, \cdots, x_T$ de palavras $w_t \in V$, na qual V representa o conjunto de palavras do vocabulário, com o objetivo de aprender um modelo $f(w_t, \cdots, w_{t-n+1}) = P(w_t|w_1^{t-1})$ sujeito a $\sum_{i=1}^{|V|} f(i, w_{t-1}, \cdots, w_{t-n+1}) = 1$. Ou seja, calcular a probabilidade da palavra w_t estar presente no contexto dado que as palavras $w_1, w_2, \cdots, w_{t-1}$ pertencem ao contexto. A função $f(w_t, \cdots, w_{t-n+1}) = P(w_t|w_1^{t-1})$ pode ser obtida decompondo-a em duas partes:

1. Realizar um mapeamento C de cada elemento i em V para um vetor real $C(i) \in \Re^m$, representando o vetor de características associado a cada palavra do vocabulário;

2. A função de probabilidade sobre as palavras, expressada por C (mapeamento realizado) : uma função g mapeia uma sequencia de entrada de vetores de características das palavras do contexto $C(w_{t-n+1}), \cdots, C(w_{t-1})$, para uma distribuição de probabilidade condicional sobre as palavras de V para a próxima palavra w_t. A saída de g é um vetor no qual o i-ésimo elemento estima a probabilidade $P(w_t = i|w_1^{t-1})$ como na Figura 2.14.

A função f corresponde à composição dos dois processos de mapeamento descritos, C e g. Cada uma das partes de f é associada a alguns parâmetros. Os parâmetros do mapeamento C são os vetores de características, representado pela matriz C de dimensões $|V| \times m$, na qual a linha i corresponde ao vetor de características $C(i)$ da palavra i. A função g pode ser implementada por meio de uma rede neural *feed-forward*.

Sucintamente, o modelo proposto por Bengio busca gerar um vetor de características denso e de tamanho reduzido para a representação de palavras. Para isso ele propõe

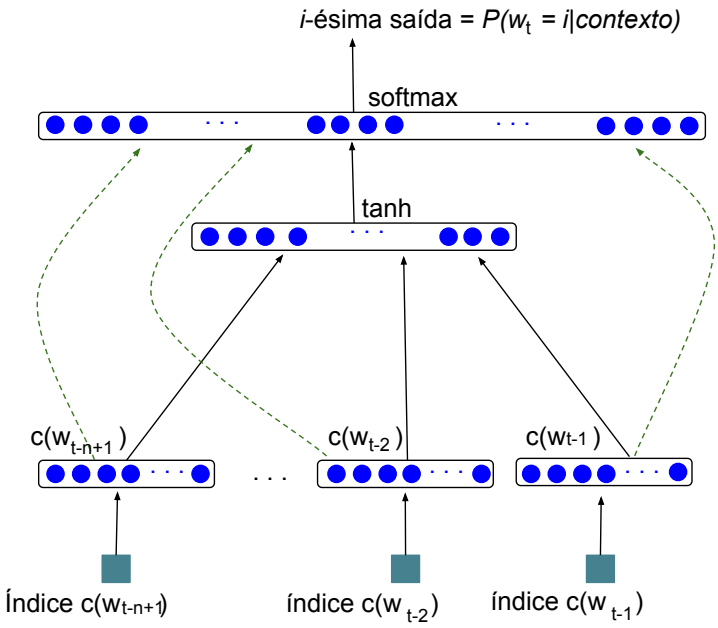


Figura 2.14: Representação neural de $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}, \cdots, C(w_{t-n+1}))$

uma rede neural a qual usa de propriedades probabilísticas para direcionar o aprendizado dos parâmetros que irão compor os vetores de características. Essas propriedades podem ser vistas como a probabilidade de uma palavra w_{i+1} ocorrer como próximo elemento do texto, dado que ela é precedida pelas palavras $w_1, w_2, \ldots, w_i$.

2.5.2 *Skip-gram* e *CBOW*

Um ponto negativo do modelo proposto por Bengio é a necessidade de se utilizar um tamanho fixo do contexto a ser analisado, devido à restrição imposta pelo modelo *feedforward*. Neste caso a rede consegue visualizar e analisar apenas cinco a dez palavras precedentes para prever a próxima, mas os seres humanos são capazes de analisar um contexto muito maior do que apenas cinco palavras. Sendo assim, Mikolov propôs a utilização de redes neurais recorrentes para esta tarefa, explorando a habilidade de memória de longo prazo, fornecida pelas redes recorrentes, como evidenciado na seção 2.3.2.

A arquitetura da rede neural recorrente utilizada por Mikolov (MIKOLOV et al., 2013b) possui uma camada de entrada x, uma camada oculta s, que também é conhecida por camada de contexto, e uma camada de saída y. A entrada da rede no tempo t é dada por $x(t)$, o estado da camada oculta é representado por $s(t)$ e a saída como $y(t)$. O vetor de entrada $x(t)$ é formado pela concatenação do vetor w, o qual representa a palavra atual

e pela saída dos neurônios de s do tempo $t-1$. Esta arquitetura pode ser visualizada na figura abaixo:

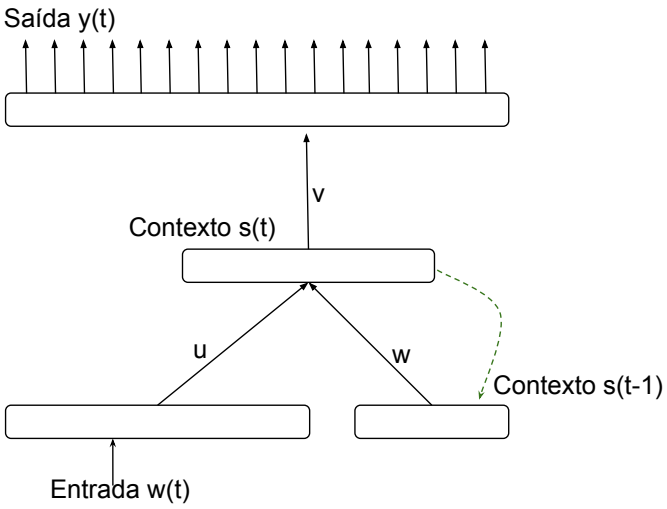


Figura 2.15: Modelo de Rede Neural Recorrente utilizado por Mikolov (MIKOLOV et al., 2013b).

É importante notar que a coluna W representa os pesos aprendidos nas iterações passadas.

As camadas de entrada, oculta e saída são calculadas como segue:

$$x(t) = w(t) + s(t-1) \tag{2.5}$$

$$s_j(t) = f(\sum_i x_i(t) u_{ji}) \tag{2.6}$$

$$y_k(t) = g(\sum_j s_j(t) v_{ki}) \tag{2.7}$$

onde $f(z)$ é a função de ativação sigmoide:

$$f(z) = \frac{1}{1+e^{-z}} \tag{2.8}$$

e $g(z)$ é a função softmax:

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \tag{2.9}$$

Ainda seguindo esta arquitetura, Mikolov propõe dois modelos para criar vetores de palavras, o *Skip-gram* e o *Continuous Bag of Words* (CBOW). O CBOW recebe como entrada uma sequência de palavras $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$ e busca prever a palavra w_i,

ou seja, é capaz de prever uma palavra dado seu contexto. Já o modelo *Skip-gram* tem como entrada uma palavra w_i e retorna as palavras $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$, ou seja, prevê o contexto dada a palavra. Uma ilustração desses modelos pode ser observada abaixo.

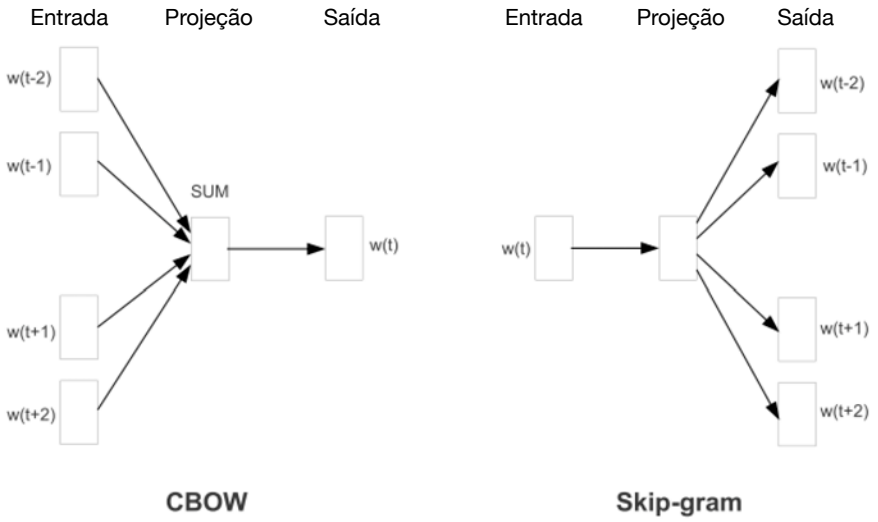


Figura 2.16: O modelo CBOW prevê uma palavra dado seu contexto, enquanto o *Skip-gram* prevê o contexto dada uma palavra. Adaptado de (MIKOLOV et al., 2013b)

De acordo com Mikolov, em um espaço vetorial as palavras semelhantes sintática ou semanticamente tendem a estarem próximas umas das outras, sendo que seu método é capaz de aprender uma representação vetorial de um grande conjunto de texto não estruturado e obter informações sobre o contexto no qual a palavra está inserida, possibilitando assim o reconhecimento de padrões linguísticos, que podem ser representados por transformações lineares ou operações algébricas, como por exemplo, o resultado do cálculo vetorial *vetor("Madri") - vetor("Espanha") + vetor("França")* é próximo à representação dada por *vetor("Paris")* (MIKOLOV et al., 2013c), i.e. o sistema aprende que "Paris" seria a melhor associação à sentença dada levando-se em consideração o contexto de *CidadeCapitalDe-País(Paris, França)*. Cabe também notar que a operação de diferença entre duas palavras é dada a partir da similaridade de cossenos. Ou seja, quanto menor o ângulo formado pelos vetores das palavras w_1 e w_2 mais próximas semântica e sintaticamente elas são.

2.5.3 *Global Vectors for Word Representation* - GloVe

Apesar de os modelos propostos por Mikolov utilizarem da teoria de probabilidade para nortearem seus processos de treinamento, eles não fazem uso eficiente da estatística (PENNINGTON; SOCHER; MANNING, 2014), pois dados do contexto global do corpus em análise não são capturados. Como o processo de criação dos vetores é concentrado

na janela dos n-gramas "deslizante" pelo corpus, o processo não opera diretamente nas estatísticas de co-ocorrência do contexto, não usufruindo da vasta informação repetida presente nos dados (PENNINGTON; SOCHER; MANNING, 2014). Um texto que trata de viagens, por exemplo, fornecerá informações sobre a palavra "viagem", "avião" e "turismo" em diversos trechos de seu conteúdo. Essas informações correlatas não são capturadas pelos modelos *Skip-gram* e *CBOW*.

Uma das principais ideias que fundamentam este método é a de que a análise da proporção da co-ocorrência das palavras tem um grande potencial para codificar algum significado. A Tabela 2.1 considerando as probabilidades de co-ocorrência das palavras "gelo" e "vapor" com algumas outras palavras do vocabulário:

Probabilidade e Proporção	$k = sólido$	$k = gás$	$k = água$	$k = moda$
$P(k\|gelo)$	$1,9 \times 10^{-4}$	$6,6 \times 10^{-5}$	$3,0 \times 10^{-3}$	$1,7 \times 10^{-5}$
$P(k\|vapor)$	$2,2 \times 10^{-5}$	$7,8 \times 10^{-4}$	$2,2 \times 10^{-3}$	$1,8 \times 10^{-5}$
$P(k\|gelo)/P(k\|vapor)$	$8,9$	$8,5 \times 10^{-2}$	$1,36$	$0,96$

Tabela 2.1: Co-ocorrência das palavras "gelo" e "vapor" com algumas outras palavras do vocabulário.

Como esperado, a palavra gelo aparece mais frequentemente com a palavra sólido, enquanto a palavra vapor com a palavra gás e com frequências similares com a palavra água, que é propriedade de ambas.

Na proporção de probabilidades, o ruído de palavras não discriminativas se cancelam (proporção próxima a 1), enquanto grandes valores (significativamente maiores que 1) se relacionam com as propriedades específicas do gelo e valores pequenos (significativamente menores que 1) com as propriedades específicas do vapor, o que indica que essa proporção de probabilidades foi capaz capturar os significados associados a conceitos da fase termodinâmica (PENNINGTON; SOCHER; MANNING, 2014).

O objetivo de treinamento do modelo GloVe é aprender os vetores de modo que o produto vetorial das palavras seja igual ao logaritmo da probabilidade de co-ocorrência das mesmas. Como o logaritmo de uma proporção é igual à diferença dos logaritmos, a função objetivo associa as proporções probabilísticas de co-ocorrência a diferenças vetoriais no espaço vetorial. Sendo assim, essa diferença é codificada no espaço vetorial como vetores de diferenças.

A Figura 2.17 ilustra duas outras relações capturadas pelo modelo. A relação *(*CEO, empresa) e de alguns superlativos. Repare que essas relações mantém um padrão em

comum.

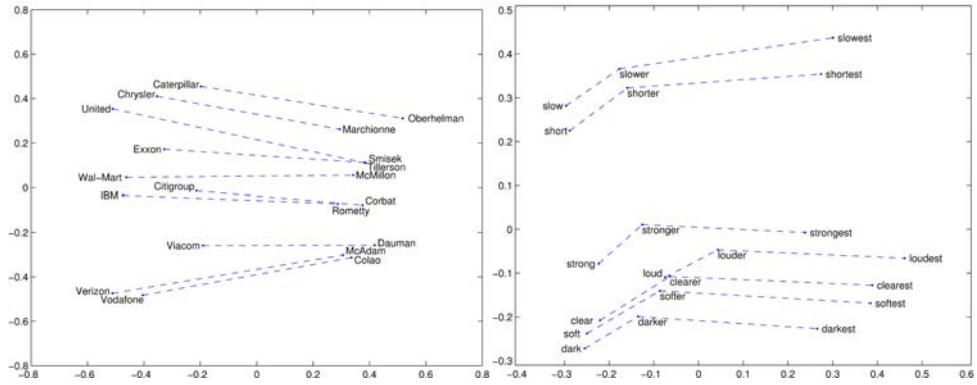


Figura 2.17: Relação de CEO com empresa e de comparativos com superlativos, capturadas pelo GloVe.

2.5.4 *Subword CBOW*

Os modelos apresentados até o momento não levam em consideração a morfologia das palavras para gerar seus respectivos vetores. Baseado no *skip-gram*, Mikolov propôs mais um modelo para a criação de *word embeddings*, denominado *Subword CBOW* (BOJANOWSKI et al., 2016). Neste modelo, os n-gramas são compostos também por caracteres, e não apenas por palavras, sendo que as palavras são representadas pela soma desses n-gramas.

Este novo modelo busca capturar as estruturas internas das palavras, o que é de grande benefício em idiomas que possuem um vasto vocabulário e um número grande de palavras raras (BOJANOWSKI et al., 2016). Uma outra vantagem deste modelo é a capacidade de lidar com palavras que não estão presentes no dicionário, ou seja, aquelas que nunca apareceram no corpus de criação dos vetores, já que ele é capaz de criar representações a nível de caracteres e que o número de caracteres presentes em qualquer idioma é consideravelmente baixo em comparação ao número de palavras e variações destas.

Durante o treinamento do *Subword CBOW*, caracteres especiais "<"e ">" são adicionados no início e fim de cada palavra. A própria palavra também é incluída na lista dos n-gramas, para que também seja aprendida a representação da palavra em si. Tomemos como exemplo a palavra *caramelo* com $n = 3$, a representação por n-gramas seria:

`<ca, car, ara, ram, ame, mel, elo, lo>` e a palavra `<caramelo>`

Cabe notar que as sequências `<ame>`, `<mel>` e `<elo>`, correspondentes às palavras *ame*, *mel* e *elo* são diferentes dos tri-gramas `ame`, `mel` e `elo`.

Suponha um dicionário e n-gramas de tamanho G. Dada uma palavra w, chamamos de $G_w \in 1,\ldots,G$ o conjunto de n-gramas presentes em w, associamos uma representação vetorial z_g a cada n-grama g e representamos uma palavra pela soma desses representações vetoriais:

$$s(w,c) = \sum_{g \in G_w} z_g^{\intercal} v_c \tag{2.10}$$

De acordo com Mikolov, esse simples modelo é capaz de compartilhar representações entre palavras, permitindo um confiável aprendizado de palavras com baixas ocorrências.

2.6 Análise de Correferência

Em muitos problemas de aprendizado de máquina são apresentadas diversas visões, descrições ou nomes para um mesmo objeto. Na linguagem escrita ou falada é comum que uma entidade ou conceito seja identificado por várias frases nominais. As situações em que duas frases nominais diferentes referem-se ao mesmo conceito são conhecidas como casos de correferência.

A tarefa de resolução de correfência consiste em determinar quais frases nominais referem-se a quais entidades em um texto. Desta forma, a correta resolução de correferências é de grande importância para tarefas de processamento de língua natural, como tradução automática de textos e extração de informações .

O problema de resolução de correferência pode ser tratado com técnicas de aprendizado supervisionado como uma tarefa de classificação, onde cada par de frases nominais possa ser classificado como correferente ou não (MCCALLUM; WELLNER, 2005). Diversos métodos supervisionados como *self-training*, *co-training* (BLUM; MITCHELL, 1998) e *Expectation Maximization* tem sido aplicados para a resolução de correferência (NG, 2008). Para (KRISHNAMURTHY; MITCHELL, 2011) um sistema capaz resolver o problema de correferêcia deve representar separadamente frases nominais, os respectivos conceitos aos quais esta frase pode se referir e a relação do tipo muitos para muitos "pode se referir a" entre eles.

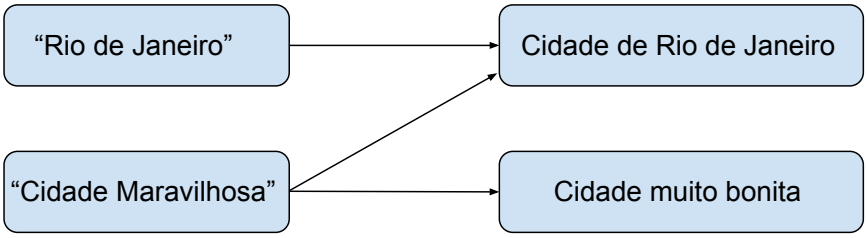


Figura 2.18: Exemplo de mapeamento de duas frases nominais para um mesmo conceito.

2.7 Classificação por Floresta de Caminhos Ótimos

A técnica de classificação baseada em floresta de caminhos ótimos modela o problema de reconhecimento de padrões como uma tarefa de partição de um grafo em um dado espaço de características, onde o vetor de características extraído de cada amostra é considerado como um nó no grafo. As arestas conectam todos os pares de nós, definindo um grafo completo. A partição do grafo é conduzida por um processo de competição entre os elementos mais representativos de cada classe, chamados de protótipos, que são obtidos durante o processo de treinamento. Os protótipos oferecem caminhos de custo ótimo às demais amostras e seus respectivos rótulos (PAPA et al., 2012; PAPA; FALCÃO; SUZUKI, 2009a). Ao final desse processo obtém-se um conjunto de treinamento particionado em árvores de caminhos ótimos, sendo que a união dessas árvores remete a uma floresta de caminhos ótimos.

Seja $\mathscr{Z} = \mathscr{Z}_1 \cup \mathscr{Z}_2 \cup \mathscr{Z}_3$ uma base de dados e $\mathscr{Z}_1$ $\mathscr{Z}_2$ e $\mathscr{Z}_3$ os conjuntos de treinamento, validação e teste, respectivamente. Seja $\mathscr{S} \in \mathscr{Z}_1$ um conjunto de protótipos de todas as classes (isto é, amostras que melhor representam as classes). Seja $\vec{v}$ um algoritmo que extrai n atributos de qualquer amostra $s \in \mathscr{Z}_1 \cup \mathscr{Z}_2 \cup \mathscr{Z}_3$ e retorna um vetor de atributos $\vec{v}(s)$ in $\Re^n$. Obtém-se a distância $d(s,t)$ entre duas amostras, s e t, como a distância entre suas amostras por meio da distância de seus vetores de características $\vec{v}(s)$ e $\vec{v}(t)$, utilizando qualquer métrica válida (Euclidiana, por exemplo).

O problema de otimização consiste em usar $\mathscr{S}$, $(\vec{v},d)$, $\mathscr{Z}_1$ e $\mathscr{Z}_2$ para projetar um classificador otimal, o qual pode-se predizer o rótulo correto $\lambda(s)$ de qualquer amostra $s \in \mathscr{Z}_3$. Assim sendo, O OPF cria partições otimais do espaço de características de forma com que todas as amostras $s \in \mathscr{Z}_2$, $\mathscr{Z}_3$ sejam classificadas de acordo com sua própria partição.

Seja $(\mathscr{Z}_1,A)$ um grafo completo cujos nós são amostras em $\mathscr{Z}_1$, onde todo par de

amostras define um arco em A (Figura 2.19b), sendo $A = \mathscr{Z}_1 X \mathscr{Z}_1$, um caminho é uma sequência de amostras $\pi = \langle s_1, s_2, ..., s_k \rangle$, onde $(s_i, s_{i+1}) \in A$ para $1 \leq i \leq k-1$. um caminho é dito trivial se $\pi = \langle s_1 \rangle$. A cada caminho π é dado o custo de uma função suave f (FALCÃO; STOLFI; LOTUFO, 2004), denotada por $f(\pi)$. Um caminho π é dito ótimo se $f(\pi) \leq f(\tau)$ para qualquer caminho τ, onde π e τ terminam na mesma amostra s, independentemente de sua origem. A concatenação do caminho π com término em s e arco (s,t)é denotada por $\pi \cdot \langle s,t \rangle$.

Apesar de o OPF poder ser utilizado com qualquer função **suave** (PAPA; FALCÃO; SUZUKI, 2009b), neste trabalho é adotada a função f_{max}, obtida como segue:

$$\begin{aligned} f_{max}(\langle s \rangle) &= \begin{cases} 0 & \text{se } s \in S, \\ +\infty & \text{caso contrário} \end{cases} \\ f_{max}(\pi \cdot \langle s,t \rangle) &= \max\{f_{max}(\pi), d(s,t)\}, \end{aligned} \tag{2.11}$$

sendo que f_{max} calcula a distância máxima entre amostras adjacentes em π, se π não é um caminho trivial, e $d(s,t)$ é a distância entre as amostras s e t no caminho π.

O algoritmo OPF associa um caminho ótimo $P^*(s)$ de $\mathscr{S}$ a toda amostra $s \in \mathscr{Z}_1$, formando uma floresta de caminhos ótimos P por meio de uma função sem ciclos, que associa a todo $s \in \mathscr{Z}_1$ seu predecessor $P(s)$ em $P*(s)$, ou marca *nil* quando $s \in \mathscr{S}$. Seja $R(s) \in S$ a raiz de $P^*(s)$, a qual pode ser alcançada usando $P(s)$. O algoritmo OPF computa, para cada $s \in \mathscr{Z}_1$ o custo $V(s)$ de $P^*(s)$, o rótulo $L(s) = \lambda(R(s))$ e seu predecessor $P(s)$, como segue no algoritmo abaixo.

Algoritmo 1 – OPF

ENTRADA: Um conjunto de treinamento $\mathscr{Z}_1$, λ-rotulado, protótipos $S \subset \mathscr{Z}_1$, par (v,d) para vetor de características e cálculo das distâncias.
SAÍDA: Floresta de caminhos ótimos P, mapa de valores de custo de caminhos V e mapa de rótulos L.
AUXILIARES: Fila de prioridades Q, e variável cst.

1. ***Para todo*** $s \in \mathscr{Z}_1$ ***Faça***
2. $\quad$ $P(s) \leftarrow nil$ *e* $V(s) \leftarrow +\infty$.
3. ***Para todo*** $s \in \mathscr{S}$, ***Faça***
4. $\quad$ $V(s) \leftarrow 0$, $P(s) \leftarrow nil$, $L(s) = \lambda(s)$ *e insira em* Q.
5. ***Enquanto*** Q *não é vazia,* ***Faça***
6. $\quad$ *Remova de* Q *uma amostra* s *tal que* $V(s)$ *é mínimo.*
7. $\quad$ ***Para cada*** $t \in \mathscr{Z}_1$, *tal que* $s \neq t$ *e* $V(t) > V(s)$, ***Faça.***
8. $\quad\quad$ *Calcule* $cst \leftarrow max\{V(s), d(s,t)\}$.

9. **Se** $cst < V(t)$, **Então**
10. **Se** $V(t) \neq +\infty$, **Então**
11. *remova* t *de* Q.
12. $P(t) \leftarrow s$, $L(t) \leftarrow L(s)$ *e* $V(t) \leftarrow cst$.
13. *Insira* t *em* Q.

As linhas 1 a 4 inicializam o algoritmo e inserem os protótipos em Q. O laço correspondente às linhas 5 a 13 calcula o caminho ótimo de $\mathscr{S}$ para cada amostra $s \in \mathscr{Z}_1$ em ordem não decrescente de custos. Um caminho de custo ótimo $V(s)$ é obtido em P a cada iteração. Quando dois caminhos atingem uma determinada amostra s com o mesmo custo mínimo, s é associado ao primeiro caminho que o atingiu. Caso o caminho que atinge uma amostra adjacente t através de s tenha menor custo que o caminho que termina em t, Q, $P(t)$, $L(t)$ e $V(t)$ são atualizados. Ao término do algoritmo V armazena o calor do custo do caminho ótimo de $\mathscr{S}$ a cada amostra $s \in \mathscr{Z}_1$ de acordo com f_{max}.

O algoritmo OPF pode ser dividido em duas fases: treinamento e teste. Durante a fase de treinamento o processo de competição é inicializado com o cálculo dos protótipos. Para isso, uma Árvore de Espalhamento Mínimo (MST) é criada sobre o grafo de treinamento original (Figura 2.19b) e os protótipos são marcados como aqueles que conectam nós de diferentes rótulos. A Figura 2.19c ilustra a MST com os protótipos (amostras circuladas) conectando diferentes classes. Removendo-se os arcos entre classes diferentes pode-se gerar então uma floresta de caminhos ótimos (Figura 2.19d). Uma classe pode ser representada por múltiplos protótipos, e deve existir pelo menos um protótipo por classe.

A fase de classificação é executada tomando-se uma amostra do conjunto de teste (triângulo da figura 2.19e) e conectando-a a todas as amostras de treinamento. A distância para todos os nós do conjunto de treinamento é utilizada como pesos das arestas. Enfim, cada nó do conjunto de treinamento oferece à amostra de teste um custo obtido pela função de custo (Equação 2.11). O nó do conjunto de treinamento que oferece o caminho de custo mínimo conquista a amostra de treinamento, como ilustrado na Figura 2.19f.

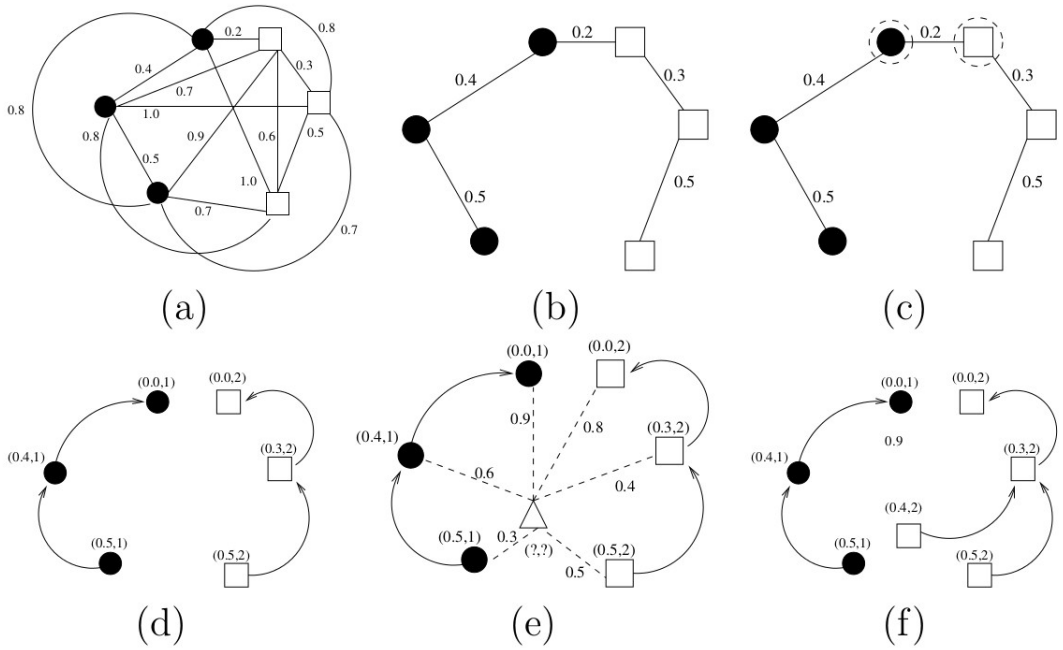


Figura 2.19: OPF pipeline

2.8 Busca Harmônica

A Busca Harmônica (*Harmony Search - HS*) é um algoritmo meta-heurístico inspirado no processo de improvisação musical utilizado por músicos (GEEM, 2009). A ideia principal consiste em modelar qualquer possível solução como uma harmonia, e cada parâmetro a ser otimizado como um instrumento musical. A melhor harmonia (solução) é escolhida como aquela que maximiza algum critério de otimização. O algoritmo é composto pelas seguintes etapas:

- Passo 1: Inicializa o problema de otimização e parâmetros do algoritmo;
- Passo 2: Inicializa a Memória Harmônica (MH);
- Passo 3: Improvisa uma nova harmonia para MH;
- Passo 4: Atualiza a MH se a nova harmonia é melhor que a pior harmonia na MH, inclui a nova harmonia na MH, e remove a pior harmonia da MH, e
- Se o critério de parada não for satisfatório, retorna para o Passo 3.

Para descrever o funcionamento da Busca Harmônica, um problema de otimização é

especificado no Passo 1, conforme segue:

$$\text{Minimizar } f(x^j) \text{ sujeito a } x^j \in X_j, \forall_j = 1,2,...,N, \tag{2.12}$$

onde $f(x^j)$ é a função objetivo, x^j e X_j, significam, respectivamente, a variável a ser otimizada j e seu conjunto de valores possíveis, e N é o número de variáveis do projeto.

Os parâmetros do algoritmo HS necessários para resolver o problema de otimização (Equação 2.12) também são especificados nessa etapa, sendo o tamanho da memória harmônica (*Harmony Memory Size - HMS*), a taxa de memória harmônica considerada (*Harmony Memory Consideration Rate - HMCR*), a taxa de ajuste (*Pitch Adjusting Rate - PAR*), e o critério de parada. Os parâmetros HMCR e PAR são usados para melhorar o vetor solução, ou seja, eles podem ajudar o algoritmo a encontrar soluções globais e locais.

2.8.1 Memória Harmônica

No Passo 2, a matriz MH é inicializada com vetores solução gerados aleatoriamente com seus respectivos valores da função objetivo:

$$MH = \left[\begin{array}{cccc|c} x_1^1 & x_1^2 & \dots & x_1^N & F(x_1) \\ x_2^1 & x_2^2 & \dots & x_2^N & F(x_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{HMS}^1 & x_{HMS}^2 & \dots & x_{HMS}^N & F(x_{HMS}) \end{array}\right], \tag{2.13}$$

onde x_i^j denota a variável de decisão da harmonia i.

2.8.2 Gerando uma Nova Harmonia

No Passo 3, um novo vetor de harmonia $\vec{x} = (\vec{x}^1, \vec{x}^2, \dots, \vec{x}^N)$ é gerado com base na considerações da memória e randomização (improvisação musical), como descrito a seguir:

$$\vec{x}^i \leftarrow \begin{cases} \vec{x}^i \in \{x_i^1, x_i^2, \dots, x_i^N\} & \text{com probabilidade HMCR,} \\ \vec{x}^i \in \mathscr{X}_i & \text{com probabilidade (1 - HMCR).} \end{cases} \tag{2.14}$$

Assim, temos que HMCR é a probabilidade de escolher um valor a partir de valores armazenados na MH, e (1-HMCR) é a probabilidade de escolher aleatoriamente um valor viável não se limitando aos armazenados na MH. Além disso, cada componente i do novo

vetor de harmonia $\vec{x}$ é examinado para determinar se ele deve ser ajustado:

$$\vec{x}^i \leftarrow \begin{cases} \textbf{Sim} \text{ com probabilidade PAR,} \\ \textbf{Não} \text{ com probabilidade (1-PAR).} \end{cases} \tag{2.15}$$

O ajuste de cada variável de decisão é frequentemente utilizado a fim de melhorar as soluções e evitar ótimos locais. Trata-se de um mecanismo que muda os valores vizinhos de algumas variáveis de decisão da harmonia. Assim, se a decisão de ajuste para a variável de decisão $\vec{x}^i$ for "Sim", $\vec{x}^i$ é substituído da seguinte forma:

$$\vec{x}^i \leftarrow \vec{x}^i + rb, \tag{2.16}$$

onde b é uma distância arbitrária, e r é uma distribuição uniforme entre 0 e 1.

2.8.3 Atualizando a Memória Harmônica

No Passo 4, se a nova harmonia for melhor do que a pior harmonia na HM, este último é substituído pela nova harmonia. Sendo a melhor harmonia aquela que maximiza a função objetivo.

2.8.4 Critério de Parada

No Passo 5, o algoritmo de Busca Harmônica finaliza sua execução quando ele satisfaz o critério de parada. Caso contrário, os Passos 3 e 4 são repetidos a fim de improvisar novamente uma nova harmonia.

2.9 Busca Harmônica auto-adaptativa

A Busca Harmônica auto-adaptativa (*Self-adaptative Harmony Search - SGHS*) (GEEM, 2008), é uma variação da Busca Harmônica que emprega um novo esquema de improvisação e parâmetro adaptativo. Esse algoritmo toma vantagem do melhor vetor de harmonias $\vec{x}_{best}$ para produzir um novo vetor $\vec{x}$. Entretanto, o ajuste de *pitch* pode alterar $\vec{x}_{best}$, de forma que $\vec{x}$ seja um vetor pior que $\vec{x}_{best}$. Portanto, para manter as boas características de $\vec{x}_{best}$, um novo ajuste de *pitch* é apresentado a seguir:

$$\vec{x}^i = x^i_{best}, i = 1, 2, \ldots, N. \tag{2.17}$$

Neste caso, $\vec{x}^i$ é atribuído de acordo com a variável correspondente x^i_{best} em $\vec{x}_{best}$. Este

novo método pode ser resumido como segue:

Algoritmo 2 – Passo 3 SGHS

1. ***Para*** $i = 0$ ***até*** N ***Faça***
2. ***Se*** $rand(0,1) \leq HMCR$, ***Então***
3. $\vec{x}^i = \vec{x}^i \pm r$, *onde* $r \in (0,1)$
4. ***Se*** $rand(0,1) \leq PAR(t)$, ***Então***
5. $\vec{x}^i = x^i_{best}$
6. ***Senão***
7. $\vec{x}^i = LB_i + r \times (UB_i - LB_i)$

onde $rand(0,1)$ retorna um valor aleatório entre 0 e 1, UB_i e LB_i são os limites superiores e inferiores de cada variável de decisão, respectivamente.

2.10 Otimização por Enxame de Partículas

A Otimização por Exame de Partículas (*Particle Swarm Optimizaton - PSO*) é um algoritmo baseado em inteligência evolutiva que encontra as soluções em um espaço de busca baseado em dinâmicas de comportamento social (KENNEDY; EBERHART, 2001). Cada possível solução do problema é modelada como uma partícula na multidão que imita sua vizinhança baseada numa função de ajuste.

Outras definições consideram o PSO como uma abordagem estocástica e um algoritmo de busca baseado em populações, onde o aprendizado do comportamento social admite que cada possível solução (partícula) possa "voar" no seu espaço (enxame) procurando por partículas que possuam as melhores características, ou seja, as que maximizam a função de ajuste. Cada partícula possui uma memória que armazena a sua melhor solução (máximo local) e a melhor solução obtida pelo enxame (máximo global).

Esse processo simula, por exemplo, um bando de pássaros procurando por comida, cujo mecanismo sócio-cognitivo pode ser resumido em três princípios (KENNEDY; EBERHART, 2001): (i) evolução, (ii) comparação e (iii) imitação. O enxame é modelado pelas partículas em um espaço de busca multidimensional $\Re^N$, onde cada partícula $p_i = (x_i, v_i) \in \Re^N$ tem duas características principais: (i) posição (x_i) e velocidade (v_i). A melhor posição local

$\hat{x}_i$ e global $\hat{s}$ também são conhecidas. Depois de definido o tamanho do enxame, ou seja, o número de partículas, elas são inicializadas com valores aleatórios de velocidade e posição. Então, cada indivíduo é avaliado de acordo com alguma função de ajuste e, caso sua solução seja melhor que a anterior, seu máximo local é atualizado. O máximo local é então atualizado com a partícula que atinge a melhor posição do enxame.

Esse processo é repetido até que o critério de convergência seja atingido. As equações de atualização de velocidade e posição de cada partícula p_i são, respectivamente, dadas por:

$$\vec{v}_i = w\vec{v}_i + c_1 r_1(\hat{x}_i - \vec{x}_i) + c_2 r_2(\hat{s} - \vec{x}_i) \tag{2.18}$$

e

$$\vec{x}_i = \vec{x}_i + \vec{v}_i \tag{2.19}$$

onde w é o peso de inércia que controla o pode de interação entre as partículas, e r_1,$r_2 \in [0,1]$ são variáveis aleatórias que dão a ideia de estocasticidade ao PSO. As constantes c_1 e c_2 são utilizadas para guiar as partículas para boas direções no enxame.

2.11 Otimização por Enxame de Glowworms

A Otimização por Enxame de *glowworm* (*Glowworm Swarm Optimizaton - GSO*) é um algoritmo de otimização baseado em enxames, no qual os agentes são chamados de *glowworms* e codificam o valor solução de sua posição corrente. Os agentes são avaliados através de uma função objetivo sobre o valor de luciferina que ele espalha para seus vizinhos (KRISHNANAND; GHOSE, 2008). Os agentes são atraídos pelos que tem maior brilho em sua vizinhança.

Um *glowworm* $g_i \in \Re^N$ considera outro *glowworm* $g_i \in \Re^N$ como seu vizinho quando $\mathbf{g}_j$ está dentro do alcance r_i de g_i, e se o nível de luciferina de $\mathbf{g}_j$ for maior que o de $\mathbf{g}_i$. Baseando-se em um mecanismo probabilístico, cada agente seleciona um vizinho com o nível de luciferina maior que o seu e se move em direção a ele. Esses movimentos permitem que os agentes formem subgrupos e movam através do espaço de busca encontrando soluções ótimas.

Os *glowworms* são inicializados com a mesma quantia de luciferina l_0, a qual é atualizada a cada iteração adicionando-se uma quantia proporcional à função de adequação de sua posição atual, e uma fração da luciferina é subtraída a fim de simular a queda de

luciferina de acordo com o tempo. A regra de atualização é dada por:

$$l_i^{t+1} = (1-\rho)l_i^t + \gamma J(\mathbf{g}_i^{t+1}) \tag{2.20}$$

onde l_i^t é a luciferina associada ao agente $\mathbf{g}_i$ no instante t, ρ é a constante de decaimento $(0 < \rho < 1)$, γ é a constante de intensificação da luciferina, e $J(\mathbf{g}_i^{t+1})$ é o valor da função objetivo do *glowworm* $\mathbf{g}_i$, no instante $t+1$.

Para um dado *glowworm* $\mathbf{g}_i$, sua probabilidade de se mover ao vizinho $\mathbf{g}_j$ é dada por:

$$p_{ij}^t = \frac{l_j^t - l_i^t}{\sum_{\mathbf{g}_k \in \eta_i^t} l_k^t - l_i^t} \tag{2.21}$$

no qual, $\eta_i^t = \{\mathbf{g}_k | d_{ik}^t < r_i^t \vee l_i^t < l_k^t\}$ é o conjunto de vizinhos do *glowworm* $\mathbf{g}_i$ no tempo t, d_{ik}^t é a distância Euclidiana entre os agentes $\mathbf{g}_i$ e $\mathbf{g}_k$ no tempo t.

Após um *glowworm* $\mathbf{g}_i$ ter escolhido seu vizinho $\mathbf{g}_j$ para seguir seu movimento, sua nova posição é dada por:

$$\mathbf{g}_i^{t+1} = \mathbf{g}_i^t + \alpha \left(\frac{\mathbf{g}_j^t - \mathbf{g}_i^t}{\|\mathbf{g}_j^t - \mathbf{g}_i^t\|} \right) \tag{2.22}$$

onde $\|.\|$ é a norma Euclidiana, e $\alpha > 0$ é o tamanho do passo de ajuste.

Seja ρ o alcance inicial da vizinhança de cada *glowworm*. O alcance da vizinhança de um agente $\mathbf{g}_i$ no tempo $t+1$ é atualizao conforme segue:

$$r_i^{t+1} = \min\{r_s, \max\{0, r_i^t + \beta(n_t - |\eta_i^t|)\}\} \tag{2.23}$$

onde β é uma contante, r_s é um sensor radial de alcance a n_t controla o número máximo de vizinhos.

2.12 Trabalhos Relacionados

Nesta seção são apresentados trabalhos já desenvolvidos cujo objetivo principal é a resolução de correferência.

Em 2011, Jayant Krishnamuryth e Tom M. Mitchell apresentaram o *ConceptResolver* (KRISHNAMURTHY; MITCHELL, 2011), o qual foi empregado como componente da NELL com o objetivo de criar conceitos e agrupar sinônimos a partir de extrações da NELL. O *ConceptResolver* parte da suposição de que se uma frase nominal possui múltiplos sentidos, estes sentidos podem ser distinguido a partir de seu contexto. Esta desambiguação pode ser resolvida analisando-se o tipo semântico das palavras por meio do

contexto em que elas estão inseridas.

Como as categorias presentes na ontologia da NELL definem um conjunto de tipos semânticos, uma frase nominal pode se referir a no máximo um conceito em cada categoria. Por exemplo, uma frase nominal, i.e. "*Apple*", pode se referir a uma **fruta** ou à **empresa** Apple®, mas não a varias empresas.

Cada sentido de uma frase nominal é representado como uma tupla, contendo a frase nominal e sua categoria, onde a categoria é utilizada como restrição durante o processo de análise de sinônimos, ou seja, duas palavras podem ser sinônimas apenas se pertencerem à mesma categoria. Para criar um sentido, o sistema interpreta cada predicado de categoria $c(x)$ de forma com que cada categoria c possua um conceito denotado pela frase nominal x. Como o *ConceptResolver* assume que existe apenas um conceito denotado por essa frase nominal, um sentido de x é criado para cada categoria extraída, Por exemplo, para uma frase que contenha ***empresa****("Apple")* e ***fruta****("Apple")*, serão criados dois sentidos para "**Apple**": *("Apple", empresa)* e *("Apple", fruta)*.

Após a extração dos sentidos de cada frase nominal, o *ConceptResolver* busca criar relações entre estes sentidos a partir das relações da entrada e das restrições definidas na ontologia. Por exemplo, a frase nominal ***CeoDaEmpresa****("Steve Jobs","Apple")* é mapeada para ***CeoDaEmpresa****(("Steve Jobs",Ceo),("Apple",Empresa))* e não seria mapeada para *("Apple",Fruta)*, pois o sentido *("Apple",Fruta)* não pertence aos tipos de argumentos aceitos por ***CeoDaEmpresa***. Este processo é eficiente pois as relações na ontologia tem domínios e argumentos restritos, sendo que apenas uma pequena parte dos sentidos gerados satisfazem essas condições de restrição.

O resultado final da etapa de indução de sentidos é uma base de conhecimento não ambígua, onde cada frase nominal foi convertida para um ou mais sentidos com relações entre pares de sentidos.

Após o mapeamento de cada frase nominal para um ou mais sentidos, o *ConceptResolver* realiza um agrupamento semi-supervisionado para encontrar sentidos sinônimos. Como apenas alguns sentidos de uma mesma categoria podem ser sinônimos, o sistema executa o processo de agrupamento para cada categoria de maneira diferente.

O *ConceptResolver* baseia-se no fato de que as relações semânticas e características da frase possibilitam uma visão independente dos dados, ou seja, é possível verificar se duas frases nominais são sinônimas analisando-se a similaridade entre os textos ou a similaridade das relações que eles compõem na ontologia da NELL. Desta forma, podemos verifi-

car que *("Empresa Apple", Empresa)* e *("Apple",Empresa)* são sinônimos porque "Empresa Apple" e "Apple" são semelhantes, como também porque aprendemos que *("Steve Jobs", Ceo)* é CEO das duas empresas nas relações ***CeoDaEmpresa****("Steve Jobs","Apple")* e ***CeoDaEmpresa****("Steve Jobs","Empresa Apple")*.

De acordo com Hruschka e Duarte (DUARTE; Hruschka Jr., 2014) uma das principais limitações do *ConceptResolver* está relacionada ao fato de a base de dados da NELL não ser completa. Desta forma, ao encontrar espaços vazios ou incompletos o método encontra dificuldades em identificar correferências, como exemplificado na Tabela 2.2.

Tabela 2.2: Instâncias de relações utilizadas como características semânticas para resolução de correferência. Adaptado de (DUARTE; Hruschka Jr., 2014)

Frase Nominal	Kobe Bryant	Kobe	Bryant	Lebron
tipoDeAtleta	Jogador de basquetebol	Jogador de basquetebol		Jogador de basquetebol
estadioDeAtleta	Staples Center		Staples Center	Arena Quicken Loans
tecnicoDeAtleta	Byron Scott		Byron Scott	David Blatt
atletaJogaNaLiga	NBA			NBA
atletaJogaParaTime		LA Lakes		Cleveland Cavaliers
atletaConhecidoComo			Kobe	Lebron James

A fim de resolver este problema, Duarte e Hruschka propõem uma abordagem híbrida para o modelo de resolução de correferência, que combina características semânticas e morfológicas extraídas das frases nominais. Sendo assim, dado um par de frase nominal *(nf1,nf2)* as características morfológicas extraídas em (DUARTE; Hruschka Jr., 2014) são:

- **Diferente Número de Palavras (DNP)**: atributo binário cujo valor é 1 (um) se *nf1* tem menos palavras que *nf2* ou vice-versa. Se as duas frases tem o mesmo número de palavras, DNP é 0 (zero). Assim, DNP(*Kobe Bryant, Kobe*) = 1 e DNP(*Bryant, Lebron*) = 0;

- **Subconjunto**: característica binária com valor 1 (um) se *nf1* é parte de *nf2*, ou *nf2* é parte de *nf1*. Caso contrário, assume valor 0 (zero). Portanto, Subconjunto(*Kobe Bryant, Kobe*) = 1, e Subconjunto(*Kobe Bryant,Lebron*) = 0;

- **Similaridade de String (SS)**: valor real normalizado entre 0 e 1 obtido utilizando-se uma simples métrica de similaridade, onde quanto mais próximo a 1, mais simi-

lares são *nf1* e *nf2*;

- **Acrônimo**: característica binária cujo valor é 1 se a *string* menor é formada pelas iniciais da maior. Por exemplo, Acronimo(*Nova York, NY*) = 1;
- **Similaridade de String de Convington(SSC)**: valor real normalizado entre 0 e 1 aplicando-se o algoritmo de Convignton (COVINGTON, 1996);
- **Proximidade**: valor real entre 0 e 1 obtido aplicando-se o algoritmo de proximidade baseado em Jaro-Winkler, disponível em LingPipe (CARPENTER,).

Duas características semânticas também foram utilizadas por eles:

- **Número de relações que compartilham o mesmo valor de instância (RelaçõesCompartilhadas)**: valor inteiro obtido pela soma de relações que compartilham o mesmo valor para *nf1* e *nf2*. Tomando-se como exemplo a Tabela 2.2, RelaçõesCompartilhadas(*Kobe Bryant, Kobe*) = 5 e RelaçõesCompartilhadas(*Kobe Bryant, Lebron*) = 2;
- **Taxa do número de relações que compartilham o mesmo valor de instância (TaxaRelaçõesCompartilhadas)**: valor real entre 0 e 1 é a taxa do número de relações que compartilham o mesmo valor de instância para as frases nominais *nf1* e *nf2* levando-se em consideração todas as relações diferentes de vazio. Ou seja, para n relações nas quais a NELL aprendeu algo para as duas frases nominais, conta-se o número de valores iguais e divide-se este valor por n. Considerando a Tabela 2.2 TaxaRelaçõesCompartilhadas(*Kobe Bryant, Kobe*) = 5/6 = 0.83 e TaxaRelaçõesCompartilhadas(*Kobe Bryant, Lebron*) = 2/3 = 0.33.

A partir da base de dados da NELL, Duarte e Hruschka construíram um conjunto de dados com 200 pares de frases nominais, as quais o *ConceptResolver* classificou como correferentes. Metade delas, ou seja, 100 pares foram classificadas corretamente (verdadeiro positivo) e a outra metade foi classificada incorretamente (falso positivo). Foram realizados três experimentos:

- **Exp1**: classificador híbrido utilizando-se características semânticas e morfológicas. Foram empregadas as oito características;
- **Exp2**: classificador semântico baseado apenas nas características semânticas;

- **Exp3**: classificador morfológico, obtido utilizando-se apenas as características morfológicas.

Os resultados obtidos em (DUARTE; Hruschka Jr., 2014) são apresentados na Tabela 2.3. Nesse estudo foi observado que em geral os experimentos que utilizaram apenas atributos morfológicos obtiveram a menor taxa de acerto, por outro lado a combinação de características semânticas e morfológicas proporcionou os melhores resultados. Os resultados apresentados evidenciam que ao utilizar ambas as características morfológicas e semânticas impacta positivamente na tarefa de resolução de correferência.

Tabela 2.3: Resultados obtidos por Duarte e Hruschka. Adaptado de (DUARTE; Hruschka Jr., 2014)

Algoritmo	Semântico	Morfológico	Matriz de Confusão				F-score
			Sim		Não		
			Correto	Errado	Correto	Errado	
LibSVM Linear	Exp.1		89	11	92	8	**0.905**
	Exp. 2		82	18	91	9	0.865
		Exp. 3	66	34	95	5	0.801
LibSVM Polinomial	Exp.1		86	14	90	10	**0.88**
	Exp. 2		82	18	90	10	0.86
		Exp. 3	57	43	100	0	0.775
Regressão Logística Bayesiana	Exp.1		94	6	86	14	**0.9**
	Exp. 2		92	8	81	19	0.865
		Exp. 3	73	27	94	6	0.833
Rede Bayesiana	Exp.1		86	14	90	10	**0.88**
	Exp. 2		82	18	90	10	0.86
		Exp. 3	57	43	100	0	0.775
Logística Simples	Exp.1		89	11	91	9	**0.9**
	Exp. 2		85	15	89	11	0.87
		Exp. 3	72	28	93	7	0.823
Regressão Logística	Exp.1		89	11	93	7	**0.91**
	Exp. 2		87	13	90	10	0.885
		Exp. 3	74	26	93	7	0.833
Naïve Bayes	Exp.1		89	11	91	9	**0.9**
	Exp. 2		94	6	86	14	**0.9**
		Exp. 3	75	25	91	9	0.829
Voted Perceptron	Exp.1		86	14	86	14	**0.86**
	Exp. 2		80	20	88	12	0.84
		Exp. 3	77	23	93	7	0.849
Random Forest	Exp.1		92	8	93	7	**0.925**
	Exp. 2		95	5	80	20	0.874
		Exp. 3	76	24	90	10	0.829
J48	Exp.1		86	14	89	11	0.875
	Exp. 2		95	5	80	20	**0.89**
		Exp. 3	73	27	92	8	0.823

Capítulo 3

Metodologia Proposta

3.1 Metodologia Proposta

Como descrito na Seção 2.6, o principal objetivo deste trabalho de mestrado é realizar uma investigação que permita se obter evidências empíricas que auxiliem na obtenção subsídios que apontem para uma tendência de aceitação (ou refutação) da seguinte hipótese de pesquisa:

Hipótese: A utilização de uma terceira visão, a qual tenha como base um conjunto de atributos extraídos de um conjunto de dados textual (*corpus*) suficientemente grande, pode contribuir na melhoria da qualidade da identificação e do tratamento de correferência na NELL, quando utilizada juntamente com as duas visões já propostas em abordagens anteriores.

Para tanto, os seguintes passos metodológicos foram executados:

1. Estudo e investigação de abordagens de extração de características a partir de grande conjuntos de dados, e identificação do estado-da-arte destas abordagens;
2. Estudo e investigação de abordagens de aprendizado de máquina que permitem a classificação multi-visão, e identificação do estado-da-arte destas abordagens;
3. Estudo, investigação e implementação da aplicação das abordagens identificadas nos itens anteriores no problema de identificação e tratamento de co-referência da NELL;
4. Estudo, investigação e implementação de possíveis melhorias e adaptações das abordagens investigadas no item anterior;
5. Análise comparativa dos resultados empíricos obtidos nos itens anteriores.

Após ter sido realizada a investigação sobre o estado-da-arte das abordagens para extração de características de grandes bases de dados (como descrito no passo metodológico 1), bem como sobre o estado-da-arte das abordagens para a classificação multi-visão, duas abordagens iniciais foram selecionadas como base para se buscar responder as seguintes perguntas:

1. Qual é o impacto da utilização de cada uma das 3 diferentes visões de maneira independente na classificação aplicada à identificação de co-referências na BC da NELL?

2. Qual é o impacto da utilização de pares (combinação dois a dois) das diferentes visões na classificação muti-visão aplicada à identificação de co-referências na BC da NELL?

3. Qual é o impacto da utilização de diferentes abordagens de classificadores multi-visão nos experimentos utilizados para responder as duas primeiras perguntas?

A utilização de combinação de diferentes visões no processo de classificação permite a união de características complementares, gerando um conjunto maior de características, e assim fornecendo mais informações ao classificador sobre os sintágmas a serem analisados, o que aumenta as evidências do modelo de classificação sobre um par de sintágmas nominais ser correferente ou não, mesmo que a NELL não tenha muita informação sobre estes sintágmas. Por exemplo, ao utilizar o par de características semânticas e morfológicas é possível analisar não apenas a estrutura de cada sintágmas, mas também sua relação com os sintágmas que compõem a mesma frase nominal.

Este trabalho foi organizado em duas fases, uma primeira visando validar a proposta de Duarte (DUARTE; Hruschka Jr., 2014) e aplicar uma metodologia de combinação de características semânticas e morfológicas empregando o classificado Floresta de Caminhos Ótimos, inspirado em (MANSANO et al., 2012); e uma segunda, que visa validar o emprego de características não supervisionadas e a aplicabilidade de modelos de redes neurais profundas para o problema de análise de correferência.

Neste trabalho foram empregadas duas bases de dados extraídas da base de conhecimentos da NELL. Uma delas, que chamamos de Base197, é a mesma empregada em (DUARTE; Hruschka Jr., 2014), composta por 200 pares de frases nominais, dos quais 100 são co-referentes e 100 são não correferentes, onde três pares foram removidos dos experimentos porque não tinham referências no modelo Skip-gram[1]. Como 197 pares é um

[1]Modelo proposto por Mikolov (MIKOLOV et al., 2013a).

número relativamente pequeno de exemplos para utilizarmos em um modelo de aprendizado profundo, foi extraída uma nova base, com 1570 pares de frases nominais, divididos igualmente entre correferentes e não correferentes, esta última denominada Base1570. Em ambas as bases, a validação de correferências foi feita manualmente.

3.2 Abordagem por Combinação de Descritores

Durante a primeira fase, geramos um novo vetor obtido da distância entre os vetores que formam o par de frases nominais no modelo do Mikolov, ou seja, a diferença entre os dois vetores. Este novo vetor foi denominado *Word2Vec*. Para avaliarmos a eficiência da abordagem de combinação de descritores, realizamos experimentos combinando todas as características e aplicando a técnica de validação cruzada em cinco testes. Usamos 30% da base de dados para treinamento, 20% para avaliação e 50% para o conjunto de teste[2]. A Tabela 3.2 expõe o desempenho do classificador OPF para o experimento com os descritores simples. Ressaltando que foram utilizados o conjunto de treinamento e avaliação em cada execução, ou seja, o conjunto de treinamento total é composto por 50% das amostras. As técnicas que retornaram os melhores resultados estão destacadas em negrito.

As abordagens *Morfológica* e *Word2Vec* apresentaram os melhores taxas de acerto, revelando que as características *semânticas*, as quais foram extraídas manualmente, não nos dão maior eficiência que as extraídas automaticamente (*Word2Vec*).

Vetor de Características	**Tx. de Acerto do OPF**
Semântico	61.40%±17.09
Morfologico	**76.84%±3.41**
Word2Vec	**75.00%±1.08**

Tabela 3.1: Eficiência do descritor simples na Base197.

Vetor de Características	**Tx. de Acerto do OPF**
Semântico	58.32%±16.12
Morfologico	**70.94%±2.21**
Word2Vec	**71.13%±1.17**

Tabela 3.2: Eficiência do descritor simples na Base1570.

A fim de encontrarmos a melhor forma de combinar os vetores de características utilizados, empregamos os algoritmos de otimização abordados no Capítulo 2 com objetivo

[2]Essas porcentagens foram escolhidas empiricamente.

de selecionar a combinação que mais reduz a taxa de erro do classificador OPF. Portanto, o vetor de características combinado é formado de acordo com a seguinte equação:

$$d_{x,y} = \sum_{i=1}^{N} \alpha_i^* \delta_{D_i}^{\beta_i^*}(s,x) \tag{3.1}$$

onde $\delta_{D_i}(s,x)$ denota a distância entre as amostras s e x usando o descritor i. E os valores α e β são encontrados pelos algoritmos de otimização, conforme ilustrado a seguir.

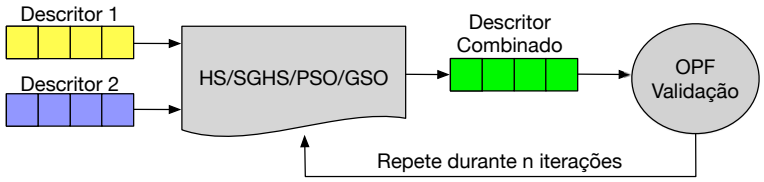


Figura 3.1: Processo de combinação de descritores. O descritor combinado é utilizado no classificador OPF sendo sua taxa de acerto a função objetivo do algoritmo de otimização empregado.

Para a tarefa de combinação de descritores foram aplicados os mesmos conjuntos de treinamento e teste que utilizamos na técnica de descritores simples, também com validação cruzada em cinco execuções, sendo 20% de todo o conjunto utilizado para avaliação. A taxa de acerto do OPF foi aplicada como função de ajuste nas técnicas PSO, GSO, HS e SGHS abordadas e os parâmetros das técnicas meta-heurísticas utilizadas foram escolhidas empiricamente como segue: número de partículas/harmonias/glowworms = 500, HMCR = 0,7, PAR = 0,7, $w = 0{,}7$, $c_1 = 1{,}6$, $c_2 = 0{,}4$, $l_0 = 5$, $r_0 = 0{,}4$. $\beta = 0{,}08$, $\gamma = 0{,}6$, $\alpha = 0{,}01$ e número de iterações = 1000.

Foram realizadas combinações de descritores por meio de todas as técnicas de otimização mencionadas. A Tabela 3.4 apresenta as taxas médias de acerto do classificador OPF para o segundo experimento. As técnicas com melhores resultados estão destacadas em negrito.

Algoritimo	**Semântico e Morfológico**	**Word2Vec e Morfológico**	**Word2Vec e Semântico**	**Todos**
PSO	81.50%±7.09	75.65%±1.37	81.82%±3.08	**82.41%±5.74**
GSO	**84.61%±1.58**	68.55%±5.54	80.43%±3.18	78.12%±4.11
HS	**84.48%±1.82**	74.38%±5.26	81.60%±2.36	81.58%±5.31
SGHS	**84.29%±1.66**	74.54%±2.50	**82.06%±1.86**	**82.02%±3.46**

Tabela 3.3: Eficiência do descritor composto na Base197.

Os resultados do segundo experimento mostram uma melhora considerável na reso-

Algoritimo	Semântico e Morfológico	Word2Vec e Morfológico	Word2Vec e Semântico	Todos
PSO	78.50%±5.11	70.97%±1.23	**78.32%±2.03**	**78.32%±5.04**
GSO	**80.29%±1.92**	64.76%±4.43	76.16%±3.58	73.97%±2.01
HS	**80.78%±1.98**	71.28%±4.76	77.47%±1.86	77.67%±3.85
SGHS	**80.19%±1.96**	69.77%±1.78	78.08%±1.96	**79.12%±2.26**

Tabela 3.4: Eficiência do descritor composto na Base1570.

lução de correferência ao utilizarmos descritores compostos. A combinação dos vetores semânticos e morfológicos apresentou uma melhora de aproximadamente 8% na taxa de acerto do OPF, além de proporcionar um menor desvio padrão, além disso, a combinação dos vetores Word2Vec e semântico obteve resultados próximos aos melhores resultados obtidos. A exceção do experimento combinando os vetores Word2Vec e morfológico, todos os outros experimentos apresentaram uma melhora entre 5% e 8%. Considerando as taxas de acerto individuais (apresentadas na Tabela 3.2) podemos concluir que a combinação de caraterísticas Word2Vec e Morfológicas pode não ser mais eficientes que seus resultados individuais, sendo que elas alcançaram resultados bastante similares. Isso pode ser explicado se considerarmos que as características Word2Vec e morfológicas podem não ser totalmente independentes, já que o Word2Vec é capaz de capturar características morfológicas do texto, como dito anteriormente.

Conforme observado na Tabela 3.2, os vetores de características extraídos por (DUARTE; Hruschka Jr., 2014) apresentam resultados significativamente melhores para análises morfológicas. Afim validar essa hipótese, a Base1570 foi coletada da NELL, em busca de características morfológicas menos evidentes entre os pares do que as presentes na Base197. Alguns dos experimentos propostos em (DUARTE; Hruschka Jr., 2014) foram executados nesta nova base a fim de compararmos com a Tabela 2.3.

Tabela 3.5: Comparação da taxa de acerto empregando-se as características utilizadas por (DUARTE; Hruschka Jr., 2014).

Algoritmo	Tx. de Acerto na Base200	Tx. de Acerto na Base1570
SVM	80,05%	67,16%
Regressão Logística	**83,50%**	65,67%
Random Forest	83,00%	**73,58%**

3.3 Abordagem com Redes Neurais

Um dos maiores benefícios do uso do *deep learning* está no auxílio à extração de características. O uso de características manualmente extraídas requer um consumo de tempo excessivo. Muitas vezes essas características são muito específicas ou incompletas para a resolução do problema, e há a necessidade de se refazer o processo para cada novo domínio. Além disso, as características empregadas no processamento de linguagem natural clássico são muito frágeis, pois dependem de representações atômicas, como identificação de pronomes, substantivos, advérbios, objeto direto, etc, criando uma estrutura rígida e sujeita a erros.

A fim de evitar os pontos negativos da extração manual de características, e após analisar o excelente desempenho das novas técnicas de representação vetorial de palavras e avanços em arquiteturas de redes neurais, foi decidido realizar o estudo dessas técnicas no problemas de resolução de correferência para este trabalho.

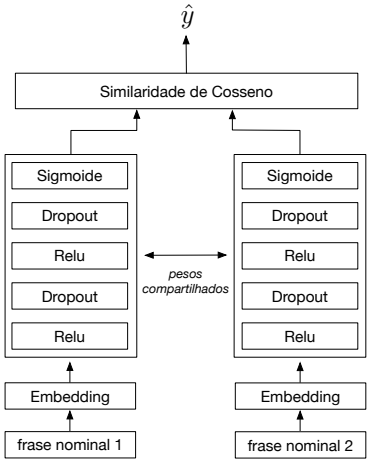
Trabalhos recentes, como os empregados em (ZAGORUYKO; KOMODAKIS, 2015) e (BERTINETTO; VEDALDI; TORR, 2017) foram propostos empregando redes profundas à arquitetura de redes siamesas, obtendo-se bons resultados. O que justifica o emprego de redes neurais siamesas neste trabalho.

Além das estruturas básicas de redes siamesas, este trabalho propõe o modelo de *Twin Siamese Network*, ou Redes Siamesas Gêmeas, onde empregamos duas redes siamesas alimentadas com vetores de características extraídos de maneiras diferentes, como *Subword CBOW* em uma e GloVe em outra, com a finalidade de combinar essas características. Uma outra abordagem foi utilizar uma Rede Neural Convolucional onde cada tipo de vetor de características compõe um canal de entrada na rede, i. e., ao empregarmos CNN em imagens podemos ter um canal para cada cor (vermelho, verde e azul).

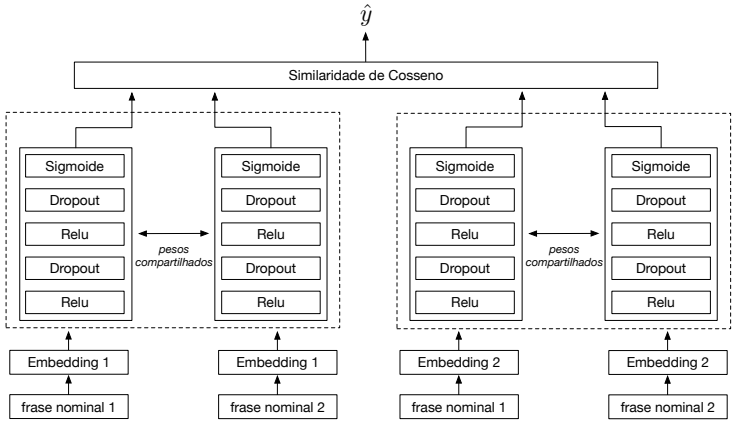
Os modelos empregados estão ilustrados na Figura 3.2.

O objetivo das redes apresentadas é mapear cada frase nominal para um espaço onde as frases não correferentes sejam linearmente separáveis de acordo com a semelhança entre os vetores que as representam.

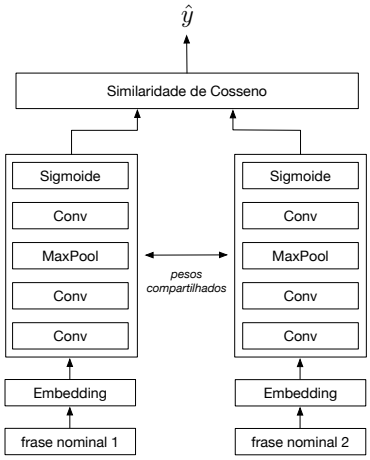
Dado que, como visto na seção 2.5, cada método de representação vetorial de palavras possui suas peculiaridades, o que inclui pontos positivos e negativos em análises semânticas e morfológicas, foram empregadas técnicas que pudessem combinar mais de um modelo vetorial para o processo de análise de correferência. Assim, as Redes Siamesas



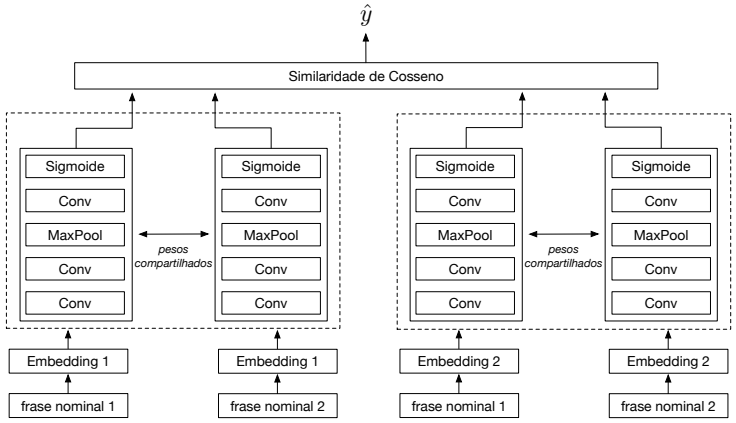
(a) Rede Neural Siamesa



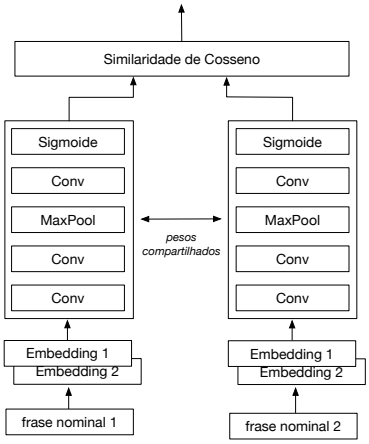
(b) Rede Siamesa Gêmea



(c) Rede Siamesa Convolucional



(d) Rede Siamesa Convolucional Gêmea



(e) Rede Siamesa Convolucional Multicanal

Figura 3.2: Redes empregadas na análise de correferência

Gêmeas são compostas de duas redes neurais siamesas, onde cada uma emprega um *word embedding* diferente para cada par de frases nominais e na última fase do processo os novos vetores, gerados por cada rede, são concatenados de acordo com sua frase nominal

correspondente antes de computar a similaridade entre o par. No caso da Rede Convolucional Multicanais, cada *word embedding* é visto como um canal diferente pertencente à frase nominal. Podemos comparar aqui a uma imagem RGB, onde o GloVe seria um nível de cor e o *Subword CBOW* seria outro.

O número de camadas empregadas nas redes foi escolhido empiricamente, considerando-se limitações de *hardware* disponível. Foram testadas de uma a cinco camadas ocultas e selecionada a configuração que apresentou o melhor resultado considerando-se também o tempo de treinamento. A fim de evitar problemas do desaparecimento do gradiente [3] foram empregadas *Rectified Linear Units*, ou ReLU, como camadas de ativação. A saída de uma ReLU é dada por:

$$f(x) = max(0,x) \tag{3.2}$$

Um comparativo do comportamento do gradiente da função ReLU e $softmax$ pode ser observado na imagem seguinte. Note que enquanto o valor do gradiente para a função $softmax$ se anula quanto mais próximo de um, o gradiente da ReLU mantém-se constante.

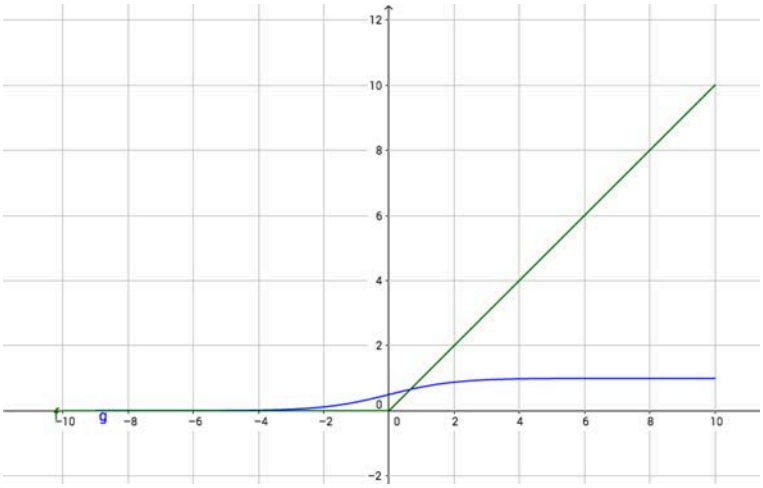


Figura 3.3: Função ReLU, representada por $f(x)$ e $softmax$ por $g(x)$.

A classificação entre pares correferentes e não correferentes é realizada tomando-se por base o conceito da álgebra vetorial, que diz que dois vetores são linearmente independentes quando seu produto vetorial é nulo, ou seja, o ângulo entre eles é 90°. Dizemos que duas frases nominais fn_1, representada pelo vetor $\vec{w}_1$, e fn_2, por $\vec{w}_2$, são correferentes se e somente se o cosseno entre os vetores $\vec{w}_1$ e $\vec{w}_2$ for igual ou inferior a 0,5. Matematicamente

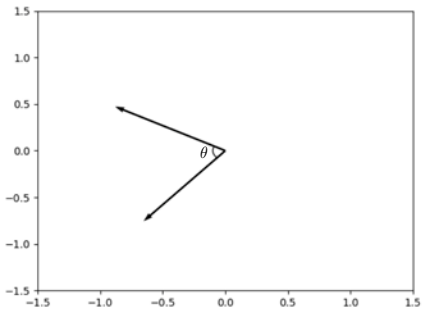
[3] Em redes neurais profundas o valor das derivadas calculadas para a obtenção do gradiente tendem a ser reduzidos conforme o número de camadas aumenta, fazendo com que a alteração dada pelo ajuste do gradiente seja quase nula.

temos:

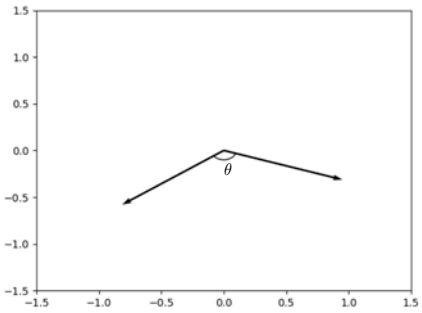
$$sim(fn_1, fn_2) = cos(\theta) = \frac{\vec{w_1} \cdot \vec{w_2}}{\|\vec{w_1}\|_2 \cdot \|\vec{w_2}\|_2} \tag{3.3}$$

onde $sim(fn_1, fn_2)$ é a função de semelhança entre as frases nominais fn_1 e fn_2 e θ é o ângulo entre suas respectivas representações vetoriais $\vec{w_1}$ e $\vec{w_2}$.

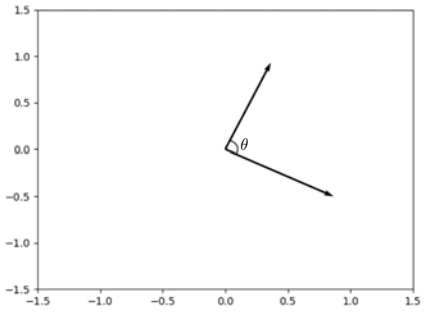
A Imagem 3.4 ilustra a representação vetorial de quatro frases nominais obtidas pelo modelo *Subword CBOW* após aplicado o algoritmo de análise de componentes principais para redução de dimensionalidade.



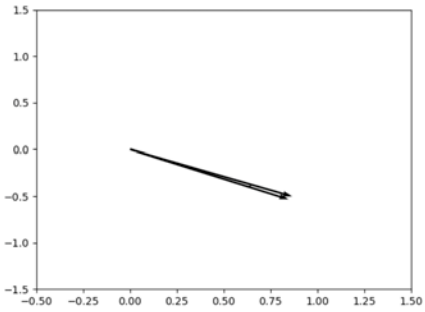
(a) *Airbus* e *British Airways*, $\theta = 60°$



(b) *Hands of Stone* e *Roberto Duran*, $\theta = 121°$



(c) *Airbus* e *British Airways*, $\theta = 86°$



(d) *Hands of Stone* e *Roberto Duran*, $\theta = 2°$

Figura 3.4: Vetores de palavras antes e após a aplicação da rede neural.

Observa-se que inicialmente o ângulo entre as palavras *Boeing* e *British Airways*, que não são correferentes, é de aproximadamente $65°$ e o ângulo entre as frases nominais *Hands of Stone* e *Roberto Duran* de $120°$. Após aplicadas à rede neural siamesa, observamos que as palavras *Boeing* e *British Airways* foram distanciadas uma da outra, formando dois vetores quase ortogonais, enquanto as frases nominais *Hands of Stone* e *Roberto Duran* estão muito próximas. A proximidade inicial entre as palavras *Boeing* e *British Airways* dá-se devido às duas estarem presentes no contexto de aviação.

Cabe ressaltar que os valor de cada dimensão dos novos vetores pertence ao intervalo $[0,1]$, a fim de evitar que haja ângulos negativos entre os vetores e normalizar o cosseno entre eles para o intervalo $[0,1]$.

Como função de *loss* foi empregada a *Contrastive loss* e, para reduzir *overfitting*, foram aplicados *droput* de 20% entre cada camada e *early-stop*.

A função de *Contrastive loss* é dada por:

$$loss = (1-y)\cdot \hat{y}^2 + y\cdot max(1-\hat{y},0)^2 \tag{3.4}$$

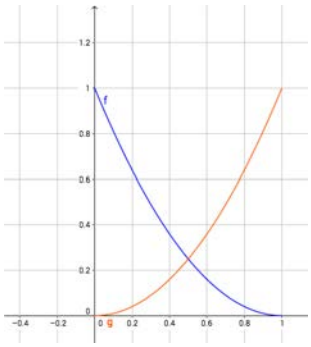


Figura 3.5: ***Contrastive loss.***

onde y é o valor real e $\hat{y}$ é o valor previsto pela rede. Caso o valor objetivo seja 1, a função de *loss* se comportará como a curva f, e se for 0 se comportará como a curva g da Imagem 3.5.

A Figura 3.6 abaixo ilustra a convergência da Rede Siamesa Simples. É possível observar que após iteração 87 a *loss* do conjunto de validação começa a aumentar, indicando *overfitting*. Nesse ponto podemos interromper o treinamento, aplicando o *early stop*.

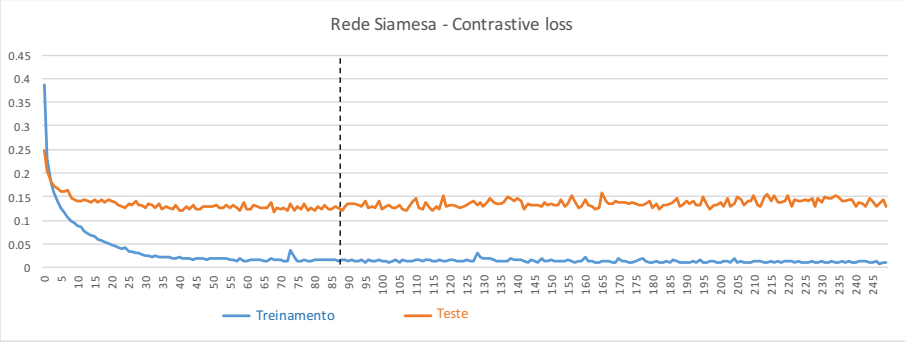


Figura 3.6: ***Convergência da Rede Neural Siamesa.***

As taxas de acerto obtidas por cada rede neural estão expostas na Tabela 3.6.

Tipo de Rede	Embedding	Tx. Acerto Treinamento	Tx. Acerto Teste
Siamesa	*Subword CBOW*	98,11%±0,39	**83,91%±1,21**
	Word2Vec	96,85%±0,91	78,92%±1,57
	GloVe	97,96%±0,03	82,94%±1,37
Siamesa Convolucional	*Subword CBOW*	98,51%±0,47	80,26%±1,11
	Word2Vec	96,07%±1,04	81,94%±1,02
	GloVe	97,87%±0,33	78,93%±0,96
Siamesa Gêmea	*Subword CBOW + Word2Vec*	98,74%±0,14	81,61%±0,93
	Subword CBOW + GloVe	98,72%±0,15	81,97%±0,73
	GloVe +*Word2Vec*	98,35%±0,12	76,61%±0,64
Siamesa Convolucional Gêmea	*Subword CBOW + Word2Vec*	98,82%±0,34	82,94%±1,31
	Subword CBOW + GloVe	98,03%±0,34	81,93%±1,31
	GloVe +*Word2Vec*	97,87%±0,24	80,60%±1,16
Siamesa Convolucional Multicanal	*Subword CBOW + Word2Vec*	98,82%±0,41	**83,94%±1,06**
	Subword CBOW + GloVe	98,78%±0,23	**84,61%±1,17**
	GloVe +*Word2Vec*	99,13%±0,17	82,94%±1,14

Tabela 3.6: Eficiência do descritor composto.

Os resultados apresentados na Tabela 3.6 mostram um aumento na taxa de acerto de aproximadamente 10% utilizando-se a rede neural Siamesa Convolucional com o *embedding Word2Vec*, se comparados aos resultados da Tabela 3.2.

Neste último experimento podemos, mais uma vez, observar que a combinação de dois descritores, ou *word embeddings*, distintos pode melhorar a capacidade de classificação para a tarefa se esses vetores de características forem independentes (e.g. extraírem características não correlacionadas), conforme pontuado na seção 3.2 e discutido por Mansano et al em (MANSANO et al., 2012). Esta abordagem permitiu uma taxa de acerto de mais de 84% para a combinação dos *embeddings Subword CBOW* e *GloVe*, que capturam, respectivamente, características morfológicas e contexto das sentenças das quais foram gerados.

No Apêndice A é possível observar os erros cometidos pelo algoritmo de classificação para um conjunto de exemplos. Observa-se que a abordagem utilizando apenas o *embedding Subword CBOW* tem uma grande concentração de erros em abreviações, isso pode se dar devido à informação local capturada pelos *embeddings* de cada palavra, sendo que uma sigla geralmente é derivada de mais de uma palavra. O *embedding GloVe*, que leva em consideração o contexto no qual as palavras estão inseridas e a co-ocorrência delas com as outras tende a gerar erros em exemplos de mesma categoria, como nos casos de "Congo" e "Rwanda", que são ambos países africanos. Por outro lado o *GloVe* fornece uma boa representação para siglas, já que, por exemplo, "U.S." e "United States" tendem a ocorrer juntos com as mesmas palavras nos corpus textuais. Sendo essas peculiaridades de grande importância para decidirmos realizar a classificação de frases nominais correferentes por meio da combinação de vetores de características distintos e complementares. O que justifica os melhores resultados obtidos através da combinação de vetores.

Capítulo 4

Considerações Finais e Trabalhos Futuros

4.1 Conclusão

Motivados pela ideia de que informações de fontes independentes podem melhorar a taxa de acerto dos algoritmos de classificação e pelo fato de que a NELL não possui muita informação sobre um sintágma nominal que está prestes a inserir em sua base de conhecimento, propomos modelar o problema de classificação de correferência como uma tarefa multivisão, por meio de técnicas que nos possibilitaram a combinação de características de fontes independentes provenientes de fora da base de conhecimento da NELL.

Por meio dos resultados obtidos podemos afirmar que as técnicas multivisão foram capazes de fornecer um aumento de aproximadamente 8% na taxa de acerto do algoritmo de classificação utilizando-se combinação de características extraídas manualmente e em mais de 10% quando utilizamos modelos mais complexos de redes neurais e diferentes tipos de *word embeddings*, automaticamente gerados a partir de um grande corpus extraído da *web*, se comparados ao uso de características individuais.

Acerca dos métodos utilizados, observa-se que os modelos de Redes Neurais empregados para a transformação espacial dos pares de sintágmas correferentes e não correferentes trouxeram melhores resultados tanto nos casos de classificação com vetores individuais quanto combinados. Porém, técnicas mais simples utilizando vetores combinados forneceram resultados melhores do que as técnicas mais complexas com vetores individuais. Sendo o melhor resultado obtido pela Rede Neural Siamesa Convolucional Multicanal com a combinação dos *embeddings Subword CBOW* e *GloVe*, capazes de capturarem características morfológicas e semânticas, respectivamente.

Desta forma, podemos concluir que a utilização de técnicas multivisão por meio de combinações de características não correlacionadas pode melhorar a taxa de acerto de um algoritmo de classificação. O que possibilita a utilização de combinação de características com baixo custo de extração com piores resultados individuais sobre a utilização de características com melhores resultados porém com alto custo de obtenção, como Mansano et al sugere em (MANSANO et al., 2012) no contexto de visualização de imagens.

4.2 Trabalhos Futuros

Técnicas mais avançadas de redes neurais tem sido amplamente utilizadas na área de processamento de linguagem natural, como as redes *Long-short Term Memory* (LSTM) (GERS; SCHMIDHUBER; CUMMINS, 2000), capazes de controlar a quantidade de memória que a rede recorrente carregará ao longo do tempo, e técnicas não supervisionadas como Autoencoders (BALDI, 2011), utilizados em ferramentas de tradução, deverão ser empregadas nos próximos passos para a classificação de correfência, em conjunto com a expansão do número de exemplos utilizados na tarefa. Também está previsto a utilização de combinação de informações de texto e imagens.

Referências

APPEL, A. P.; Hruschka Jr, E. R. Prophet – a link-predictor to learn new rules on nell. In: *2011 IEEE 11th International Conference on Data Mining Workshops*. [S.l.: s.n.], 2011. p. 917–924. ISSN 2375-9232.

BALDI, P. Autoencoders, unsupervised learning and deep architectures. In: *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*. JMLR.org, 2011. (UTLW'11), p. 37–50. Disponível em: <http://dl.acm.org/citation.cfm?id=3045796.3045801>.

BELLET, A.; HABRARD, A.; SEBBAN, M. *A Survey on Metric Learning for Feature Vectors and Structured Data*. 2013.

BENGIO, Y. et al. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, v. 3, p. 1137–1155, 2003. ISSN 15324435.

BERTINETTO, L.; VEDALDI, A.; TORR, P. H. S. Fully-Convolutional Siamese Networks for Object Tracking. 2017.

BLUM, A.; MITCHELL, T. Combining Labeled and Unlabeled Data with Co-Training. *11th Annnual Conference on Computational Learning Theory*, p. 92–100, 1998.

BOJANOWSKI, P. et al. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

BROMLEY, J. et al. Signature verification using a "siamese"time delay neural network. In: COWAN, J. D.; TESAURO, G.; ALSPECTOR, J. (Ed.). *Advances in Neural Information Processing Systems 6*. Morgan-Kaufmann, 1994. p. 737–744. Disponível em: <http://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neural-network.pdf>.

CARLSON, A. et al. Toward an architecture for never-ending language learning. In: *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*. [S.l.: s.n.], 2010.

CARPENTER, B. Lingpipe for 99.99% recall of gene mentions. In: *Proceedings of the 2nd BioCreative workshop*. Valencia, Espanha: [s.n.].

COVINGTON, M. A. An algorithm to align words for historical comparison. *Comput. Linguist.*, MIT Press, Cambridge, MA, USA, v. 22, n. 4, p. 481–496, dez. 1996. ISSN 0891-2017. Disponível em: <http://dl.acm.org/citation.cfm?id=256329.256333>.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, Springer London, v. 2, n. 4, p. 303–314, dez. 1989. ISSN 0932-4194. Disponível em: <http://dx.doi.org/10.1007/BF02551274>.

DUARTE, M. C.; Hruschka Jr., E. R. Exploring two Views of Coreference Resolution in a Never-Ending Learning System. In: . [S.l.: s.n.], 2014. p. 273–278. ISBN 9781479976331.

ELMAN, J. L. Finding structure in time. *Cognitive Science*, v. 14, n. 2, p. 179–211, 1990. Disponível em: <http://dblp.uni-trier.de/db/journals/cogsci/cogsci14.htmlElman90>.

FALCÃO, A.; STOLFI, J.; LOTUFO, R. The image foresting transform: Theory, algorithms, and applications. *IEEE Transactions on PAMI*, v. 26, n. 1, p. 19–29, 2004.

GEEM, Z. W. Novel derivative of harmony search algorithm for discrete design variables. *Applied Mathematics and Computation*, v. 199, p. 223–230, 2008.

GEEM, Z. W. *Music-Inspired Harmony Search Algorithm: Theory and Applications*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 364200184X, 9783642001840.

GERS, F. A.; SCHMIDHUBER, J. A.; CUMMINS, F. A. Learning to forget: Continual prediction with lstm. *Neural Comput.*, MIT Press, Cambridge, MA, USA, v. 12, n. 10, p. 2451–2471, out. 2000. ISSN 0899-7667. Disponível em: <http://dx.doi.org/10.1162/089976600300015015>.

GOMIDE, F. A. Redes neurais artificiais para engenharia e ciÃaplicadas: curso prÃ¡tico. *Sba: Controle AutomaÃ§Ã Sociedade Brasileira de Automatica*, scielo, v. 23, p. 649 – 652, 10 2012. ISSN 0103-1759. Disponível em: <http://www.scielo.br/scielo.php?script=sci$_arttextpid = S0103 - 17592012000500011nrm = iso$>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning. In: _____. [S.l.]: MIT Press, 2016. cap. Deep Forward Network, p. 167.

HAYKIN, S. S. *Neural networks and learning machines*. Third. Upper Saddle River, NJ: Pearson Education, 2009.

HEBB, D. O. *The Organization of Behavior: A Neuropsychological Theory*. New ed. New York: Wiley, 1949. Hardcover. ISBN 0805843000. Disponível em: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0805843000>.

HUBEL, D. H.; WIESEL, T. N. Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology (London)*, v. 195, p. 215–243, 1968.

KENNEDY, J.; EBERHART, R. C. *Swarm Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. ISBN 1-55860-595-9.

KOCH, G.; ZEMEL, R.; SALAKHUTDINOV, R. Siamese neural networks for one-shot image recognition. In: . [S.l.: s.n.], 2015.

KRISHNAMURTHY, J.; MITCHELL, T. M. Which noun phrases denote which concepts? In: *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*. [s.n.], 2011. p. 570–580. Disponível em: <http://www.aclweb.org/anthology/P11-1058>.

KRISHNANAND, K. N.; GHOSE, D. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intelligence*, v. 3, n. 2, p. 87–124, 2008. ISSN 1935-3820. Disponível em: <http://dx.doi.org/10.1007/s11721-008-0021-5>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.

MANSANO, A. et al. Improving image classification through descriptor combination. *Brazilian Symposium of Computer Graphic and Image Processing*, p. 324–329, 2012. ISSN 15301834.

MCCALLUM, A.; WELLNER, B. Conditional models of indetity uncertainty with application to noun correference. In: . [S.l.: s.n.], 2005.

MCCULLOCH, W. S.; PITTS, W. H. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, p. 115–133, 1943.

MIKOLOV, T. et al. Distributed Representations of Words and Phrases and their Compositionality. In: *Conference on Neural Information Processing Systems*. [S.l.: s.n.], 2013. p. 1–9. ISSN 10495258.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. Disponível em: <http://arxiv.org/abs/1301.3781>.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

MIKOLOV, T. et al. Recurrent Neural Network based Language Model. *Interspeech*, n. September, p. 1045–1048, 2010.

MITCHELL, T. M. et al. Never-Ending Learning. *AAAI Conference on Artificial Intelligence*, p. 2302–2310, 2015. Disponível em: <http://www.cs.cmu.edu/ xinleic/papers/aaai15.pdf>.

NG, V. Unsupervised Models for Coreference Resolution. *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, n. October, p. 640–649, 2008.

PAPA, J. P. et al. Efficient supervised optimum-path forest classification for large datasets. *Pattern Recognition*, Elsevier Science Inc., New York, NY, USA, v. 45, n. 1, p. 512–520, 2012.

PAPA, J. P.; FALCÃO, A. X.; SUZUKI, C. T. N. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, John Wiley & Sons, Inc., New York, NY, USA, v. 19, n. 2, p. 120–131, 2009. ISSN 0899-9457.

PAPA, J. P.; FALCÃO, A. X.; SUZUKI, C. T. N. Supervised pattern classification based on optimum-path forest. *Int. J. Imaging Syst. Technol.*, John Wiley & Sons, Inc., New York, NY, USA, v. 19, n. 2, p. 120–131, jun. 2009. ISSN 0899-9457. Disponível em: <http://dx.doi.org/10.1002/ima.v19:2>.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: *EMNLP*. [S.l.: s.n.], 2014. v. 14, p. 1532–1543.

QUINLAN, J. R. Learning logical definitions from relations. *MACHINE LEARNING*, v. 5, p. 239–266, 1990.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, n. 6, p. 386–408, 1958.

TORRES, R. S. et al. A genetic programming framework for content-based image retrieval. *Pattern Recognition*, Elsevier Science Inc., New York, NY, USA, v. 42, n. 2, p. 283–292, 2009. ISSN 0031-3203.

XING, E. P. et al. Distance metric learning, with application to clustering with side-information. In: *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 15*. [S.l.]: MIT Press, 2003. p. 505–512.

ZAGORUYKO, S.; KOMODAKIS, N. Learning to compare image patches via convolutional neural networks. *CoRR*, abs/1504.03641, 2015. Disponível em: <http://dblp.uni-trier.de/db/journals/corr/corr1504.htmlZagoruykoK15>.

Apêndice A

Erros cometidos pelos algoritmos de classificação

A lista abaixo apresenta os erros cometidos pela Rede Siamesa empregando-se o *Subword CBOW*, sendo FP referente a erros do tipo falso positivo e FN a falso negativo:

Frase Nominal 1	Frase Nominal 2	Tipo Erro
cisco	CSCO	FN
microsoft	MSFT	FN
Apple	Apple Inc	FN
quds force	irgc qf	FN
Rim Blackberry	RIMM	FP
Apple Inc	Apple Company	FN
Motorola	MOT	FN
OS	Oakland	FN
Research in Motion Limited	RIMM	FP
hhs	Department Health and Human Services	FN
Wayne State	state university	FN
Kent State	state university	FN
Maritime Administration	MARAD	FN
Case Western Reserve University	CWRU	FN
Michigan State University	MSU	FN
Office of Environmental Management	OEM	FP
NL	National Hockey League	FP
ITA doc	International Trade Administration	FN

Baylor	state university	FN
state university	New York University	FP
state university	Northern Arizona University	FN
state university	Johns Hopkins University	FP
BMY	Micron Technology Inc	FP
INTC	Wal Mart Stores	FP
Rwanda	Congo	FP
Chechnya	Russia	FP
Greenland	America	FP
SSS	National Women Hockey League	FP
FHF Board	FMCS Administration	FP
ADM	Bill Mcallister	FP
Aaron Mckie	PJ Carlesimo	FP
EMI	Tricky Dick	FP
FSA	Commodity Futures Trading Commission	FP
ACDA	DISA	FP
DEC	Northwestern State Lady Demons	FP
Department of Justice	Bureau of Transportation Statistics	FP
Bruins	British Airways	FP
Theodor Kollek	EEOC	FP
CBA	BTS	FP
Remy MA	ACDA	FP
Warner Music	NRA	FP
AMS	National Science Foundation	FP
Department of Commerce	UL	FP
IMLS	Advisory Council on Historic Preservation	FP
National Broadcasting Corporation	Washington State	FP
National Council on Disability	ASA	FP
Automated Clearing House	NTIS	FP
Directv	MILS	FP
Department of the Interior	Abdul Rahim Noor	FP

A lista abaixo apresenta os erros cometidos também pela Rede Siamesa, porém empregando-se o *GloVe*:

Frase Nominal 1	Frase Nominal 2	Tipo Erro
microsoft	MSFT	FN
Cisco Systems Inc	CSCO	FN
CNN	Cable News Network	FN
Rim Blackberry	RIMM	FP
Motorola	MOT	FN
OS	Oakland	FN
Research in Motion Limited	RIMM	FP
Wayne State	state university	FN
Kent State	state university	FN
Maritime Administration	MARAD	FN
Case Western Reserve University	CWRU	FN
Michigan State University	MSU	FN
United National Congress	UNC	FN
National Academy of Sciences	NAS	FN
Office of Environmental Management	OEM	FP
IBM	International Business Machines Corp	FN
UNC	United National Congress	FN
Divisio	United States Department	FP
ITA doc	International Trade Administration	FN
Members	Armed Forces	FP
Baltimore Colts	Baltimore Ravens	FP
Baltimore Ravens	Baltimore Colts	FP
Baylor	state university	FN
state university	New York University	FP
RAMS	Seahawks	FP
state university	Johns Hopkins University	FP
state university	Kansas Wesleyan University	FP
Chubb Corp	CB	FN
BMY	Micron Technology Inc	FP
INTC	Wal Mart Stores	FP
News Corp	NWS	FN
Bristol Myers Squibb	BMY	FN

JP Arencibia	JP Losman	FP
Kimora Lee Simmons	Kimora Lee Perkins	FP
Albert Belle	Albert Pujols	FP
David Coleman Headley	Daood Gilani	FN
Jamaal Magloire	Jamaal Tinsley	FP
JP Ricciardi	JP Losman	FP
Marquis Daniels	Caron Butler	FP
Rudolf Wanderone	Minnesota Fats	FP
Hank Aaron	HA	FN
Tyson Chandler	Nenad Krstic	FP
Jeff Green	Kevin Durant	FP
State University	Utah State	FN
District	Columbia	FP
Yugoslavia	Croatia	FP
Congo	Sudan	FP
Sudan	Congo	FP
Chechnya	Russia	FP
SSS	National Women Hockey League	FP
FHF Board	FMCS Administration	FP
Aaron Mckie	PJ Carlesimo	FP
EMI	Tricky Dick	FP
FSA	Commodity Futures Trading Commission	FP
ACDA	DISA	FP
Department of Justice	Bureau of Transportation Statistics	FP
Constantinople	Education Department	FP
Digital Equipment Corporation	Siebel Systems Inc	FP
Joint Chiefs of Staff	Motorola	FP
Theodor Kollek	EEOC	FP
CBA	BTS	FP
Remy MA	ACDA	FP
AMS	National Science Foundation	FP
Department of Commerce	UL	FP
IMLS	Advisory Council on Historic Preservation	FP
National Council on Disability	ASA	FP
Automated Clearing House	NTIS	FP

Sun Devils	Markit	FP
NASCAR	Members	FP
Directv	MILS	FP
Department of the Interior	Abdul Rahim Noor	FP

A lista abaixo apresenta os erros cometidos pela Rede Siamesa Multicanal, empregando-se a combinação do *Subword CBOW* e *GloVe*:

Frase Nominal 1	Frase Nominal 2	Tipo E
cisco	CSCO	FN
microsoft	MSFT	FN
Apple	Apple Inc	FN
Cisco Systems Inc	CSCO	FN
CNN	Cable News Network	FN
quds force	irgc qf	FN
quds force	quds army	FN
Apple Inc	Apple Company	FN
Wal Mart stores	WMT	FN
UL	Underwriters Laboratories	FN
ITA doc	International Trade Administration	FN
Baylor	state university	FN
Jacksonville Jaguars	Jags	FN
state university	Boston University	FP
state university	New York University	FP
state university	Northern Arizona University	FN
state university	Kansas Wesleyan University	FP
Chubb Corp	CB	FN
Kroger Co	KR	FN
Bristol Myers Squibb	BMY	FN
Kimora Lee Simmons	Kimora Lee Perkins	FP
Honest Abe	Abraham Lincoln	FN
Thomas Meskill	Tough Tommy	FN
Nashik	Nasik	FN
Jakarta	Batavia	FN
Cisco Systems	CSCO	FN
Automatic Data Proc	ADP	FN
Exxon Mobil	XOM	FN
State University	Utah State	FN
Barclays PLC Ads	BCS	FN
Technical Assistance Cooperation Committee	TACC	FN
FSA	Commodity Futures Trading Commission	FP

ACDA	DISA	FP
CBA	BTS	FP
Department of Commerce	UL	FP
National Council on Disability	ASA	FP
Major Indoor Soccer League	Colts	FP